METHOD OF EVOLVING JUNCTIONS: A NEW APPROACH TO PATH PLANNING AND OPTIMAL CONTROL

A Thesis
Presented to
The Academic Faculty

by

Jun Lu

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Mathematics

Georgia Institute of Technology
May 2014

METHOD OF EVOLVING JUNCTIONS: A NEW APPROACH TO PATH PLANNING AND OPTIMAL CONTROL

Approved by:

Professor Haomin Zhou, Advisor
School of Mathematics
Georgia Institute of Technology

Professor Magnus Egerstedt
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Shui-Nee Chow
School of Mathematics
Georgia Institute of Technology

Professor Sung Ha Kang
School of Mathematics
Georgia Institute of Technology

Professor Luca Dieci
School of Mathematics
Georgia Institute of Technology

Date Approved: March 31

To

My Parents and *Love Yixian Chen*

PREFACE

This thesis is based on several papers that have been published [17, 18, 41], submitted [19] or currently under investigation from August 2009 to present. People who collaborated are Prof. Shui-Nee Chow, Prof. Magnus Egerstedt, Prof. Haomin Zhou and Yancy Diaz-Mercado. All chapters except Chapter II (Mathematical preparation) are done solely by the collaborators list above.

Intermittent Diffusion, Section 2.3 is one critical component to the development of this thesis. In order to analyze the complexity of the algorithm, which is not included in the original paper [16], I bluntly copied the proof of the convergence from there.

Chapter I gives an introduction to the problem addressed and an overview of the method of evolving junctions developed in this thesis.

Chapter II provides necessary mathematical background for the development. Optimal control theory is familiar to engineers, but not necessarily to mathematicians. Section 2.2 defines the shortest path problem in a metric space mathematically and Section 2.3 is a moderate introduction to intermittent diffusion, a newly developed global optimization strategy.

Chapter III is the heart of this thesis. It provides the complete description of the method in a rather general setting. It will be referred in all the following chapters.

Chapter IV is based on [17, 18, 19]. They apply the method of evolving junctions into different settings, namely, $\mathbf{R}^2$ with obstacles that have smooth boundaries, $\mathbf{R}^3$ with polyhedra obstacles, and $\mathbf{R}^2$ with disk movers, shortest path between two sets dynamic obstacles.

Chpater V is based on [41], a conference paper just got accepted. It reports the real-world application of the method into robotics. It is a collaboration with Georgia

Robotics and InTelligent Systems Laboratory.

Chapter VI is based on a working paper. It addresses the shortest path problem in a dynamic environment.

ACKNOWLEDGEMENTS

This thesis would not have been possible without the support and help from the kind people around me, to only some of whom it is possible to give particular mention here.

First, I would like to express my deepest gratitude to my advisor Haomin Zhou for his guidance of the entire project, patience and encouragement. He is the ideal and perfect advisor on every aspect. I'm most grateful to him for leading me into the exciting research area and the thousands of extensive rewarding discussions. His meticulous caring and support has made my five years of research life productive, fulfilled and most importantly, enjoyable. I thank him for being a life mentor, which enables me to concentrate on this thesis. In particular, I owe him the understanding, advices, and tremendous help on my future careers. I can't sufficiently express my thanks and I couldn't imagine a better advisor or mentor in the world.

Special thanks to Prof. Shui-Nee Chow. I am indebted to him for many discussions and some sharp perspectives on the difficulties I encounter. He is the one who is most knowledgable and sees the big picture. I'm also grateful to him for several interesting stories in the scientific community, among which, the publication of the famous paper *Period Three implies Chaos* is my favorite.

I would like to thank Prof. Magnus Egerstedt and Yancy Diaz-Mercado for the enjoyable interdisciplinary collaborations. The collaboration not only gives the opportunity to see the work in real-world application, but also points out new directions for my research.

I'm extremely grateful to Prof. Luca Dieci and Sung Ha Kang for serving as my thesis committee members. I owe thanks to them for the insightful questions and

suggestions in my research during the oral exam.

Among my fellow graduate students and friends, I would like to thank Ke Yin, Wuchen Li, Weizhe Zhang, Xiaojin Ye, Yunlong He, Yao Li and Lei Zhang for discussions on many interesting problems, both in academia and in life. Also special thanks to Ruidong Wang, Lili Hu, Xiaolin Wang and Jingfang Liu for helping me through tedius things when I got lazy.

Last, but by no means least, I convey my deep thanks to my parents and love Yixian Chen, without whose constant support and caring, I wouldn't have gone so far.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

This thesis proposes a novel and efficient method (Method of Evolving Junctions) for solving optimal control problems with path constraints, and whose optimal paths are separable. A path is separable if it is the concatenation of finite number of subarcs that are optimal and either entirely constraint active or entirely constraint inactive. In the case when the subarcs can be computed efficiently, the search for the optimal path boils down to determining the junctions that connect those subarcs. In this way, the original infinite dimensional problem of finding the entire path is converted into a finite dimensional problem of determine the optimal junctions. The finite dimensional optimization problem is then solved by a recently developed global optimization strategy, intermittent diffusion. The idea is to add perturbations (noise) to the gradient flow intermittently, which essentially converts the ODE's (gradient descent) into a SDE's problem. It can be shown that the probability of finding the globally optimal path can be arbitrarily close to one. Comparing to existing methods, the method of evolving junctions is fundamentally faster and able to find the globally optimal path as well as a series of locally optimal paths. The efficiency of the algorithm will be demonstrated by solving path planning problems, more specifically, finding the optimal path in cluttered environments with static or dynamic obstacles.

CHAPTER I

INTRODUCTION

Optimal control, also known as trajectory optimization, seeks to determine the input (control) to a dynamical system that optimize a given performance functional (maximize profit, minimize cost, etc), while satisfying different kinds of constraints:

$$\text{(Dynamical System)} \qquad \dot{x} = f(x(t), u(t), t), \quad t_0 \le t \le t_f, \quad x(t) \in \mathbf{R}^n,$$

$$\text{(Performance Index)} \qquad J = \psi(x(t_f), t_f) + \int_{t_0}^{t_f} L(x(t), u(t), t)\,\mathrm{d}t.$$

Here $u(t) \in \mathbf{R}^m$ is the control, $L\colon \mathbf{R}^n \times \mathbf{R}^m \times \mathbf{R}^+ \to R$ is the running cost, also called the Lagrangian and $\psi\colon \mathbf{R}^n \times \mathbf{R}^+ \to \mathbf{R}$ is the terminal cost. t_f is the terminal time which may be undetermined. One can also impose various constraints on the path or control, for example, to require the final position $x(t_f)$ being fixed or the whole path outside certain region.

Because many engineering problems can be formulated into the framework of optimal control, the optimal control theory has vast applications. However, due to the complexity of most applications, few of them can be solved analytically. Thus numerical methods are often employed to solve them instead. Traditionally, numerical methods for optimal control problems can be divided into three categories, (1) state-space, (2) indirect, and (3) direct methods. *State-space* approaches apply the principle of optimality which states that each subarc of the optimal trajectory must be optimal. This leads to the well-known Hamiltonian-Jacobi-Bellman (HJB) equation which dates back to 1950s [5]. *Indirect methods* employs the necessary condition of optimality known as Pontryagin Maximum Principle [54]. This leads to a boundary value problem which is then solved by numerical methods. Thus this approach is often referred to as "first optimize, then discretize". The boundary value problem

is often solved by shooting techniques or by collocations, for example, neighboring extremal algorithm, gradient algorithm, quasilinearization algorithm [4, 6, 40, 44]. However, due to strong nonlinearity and instability, the above differential equations are often difficult to solve. *Direct methods* are often referred to as "first discretize, then optimize". As the name suggests, direct methods convert the original continuous infinite dimensional control problem into a finite dimensional optimization problem. This transformation is achieved by, for example, approximating the original control by piecewise constant controls. The resulting discrete problem is a standard nonlinear programming problem (NLP) which can be solved by many well established algorithms such as Newton's method, Quasi-Newton methods [21, 25, 27, 49]. The advantages of direct methods over indirect method is that the path constraints can be handled automatically by solving NLP problem. Direct methods are nowadays the most widespread and successfully used techniques.

Our aim in this thesis is to provide fast numerical methods for solving optimal control problems with path constraints, in which the optimal trajectory exhibits certain structures, known as separability. Simply put, a path $\gamma\colon [0,T] \to \mathbf{R}^n$ is said to be separable, if there exits finite number of points, called junctions

$$(\tilde{x}_0, , \tilde{x}_1, \cdots, \tilde{x}_{n+1}), \quad \tilde{x}_i = (t_i, x_i) \in \mathbf{R}^+ \times \mathbf{R}^n \tag{1}$$

such that γ can be represented as

$$\gamma_0(\tilde{x}_0, \tilde{x}_1) \cdot \gamma_c(\tilde{x}_1, \tilde{x}_2) \cdot \gamma_0(\tilde{x}_2, \tilde{x}_3) \cdot \gamma_c(\tilde{x}_3, \tilde{x}_4) \cdots \gamma_0(\tilde{x}_n, \tilde{x}_{n+1}) \tag{2}$$

where $\gamma_0(\tilde{x}_i, \tilde{x}_{i+1})$ is the optimal path connecting $\tilde{x}_i$ and $\tilde{x}_{i+1}$ with inactive constraint and $\gamma_c(\tilde{x}_i, \tilde{x}_{i+1})$ is the optimal path connecting $\tilde{x}_i$ and $\tilde{x}_{i+1}$ with active constraint and $\gamma_1 \cdot \gamma_2$ is the concatenation of two paths.

The significance of being separable is that the determination of the entire path boils down to the determination of only finite number of junctions and the determination of finite number of optimal paths of smaller size, namely, γ_0 and γ_c. On

other hand, in many application, γ_0 and γ_c can be computed either analytically or numerically very easily. Thus, in this way, the original infinite dimensional problem of finding the whole path is converted into a finite dimensional problem. One gains a tremendous dimension reduction.

Another way to see the advantage of the dimension reduction is to compare it with the NLP approach. The NLP approach discretizes the entire path and tries to find the optimal location of each grid point. The number of variables depends on the step size of the discretization. To obtain a decent accuracy, the number of variables must be relatively large. However, the separability of the optima path implies that only some of the grid points, i.e. the junctions are important and enough to determine the whole path. Since the number of junctions is finite and often bounded, to improve the accuracy, there is no need to refine the discretization any more. As one may imagine, the cost saved by taking advantage of separability is inversely proportional to the accuracy one wants to achieve, which is extremely large.

The resulting finite dimensional optimization problem can be handled by many established algorithms, for example, gradient descent. However, it is evident from many applications that the number of local minimizers is often very large. Therefore, it is critical that any useful method must be capable of obtaining the globally optimal path. In this thesis, we will adopt a recently developed global optimization strategy, called intermittent diffusion [16]. The idea is to add noise (diffusions) to the gradient flow intermittently. When the noise is turned off, one gets a pure gradient flow and it quickly converges to a local minimizer. On the other hand, when the noise is turn on, the perturbed flow will have certain chance to jump out of the local trap and converge to other local minimizers, including the global one. It can be shown, the local minimizers obtained will include the global one with probability arbitrarily close to 1.

We call the method outlined above *Method of Evolving Junctions*. The method

is able to overcome several critical drawbacks of the aforementioned three methods. Namely, HJB approach suffers from the notorious problem known as "curse of dimensionality"; Both indirect methods and directs space are only able to yield local minimizers. In addition, all three methods are all grid bases. This prevents them from giving arbitrarily accurate solutions.

Table 1: A comparison between different methods.

Methods	Local/Global	Approaches
HJB	Global	PDEs
Indirect	Local	Boundary Value ODEs
NLP	Local	Constrained NLP
Evolving Junctions	Global (Probabilistic)	SDEs

The method of evolving junctions works for any separable optimal control problems. And the only requirement for the method to be applicable is the separability of the optimal path. This is usually not a trivial task and involves different proof techniques in different settings. In this thesis, we will focus our attention on a particular type of optimal control problems, i.e. *path planning problems*. More specifically, we would like to find the shortest or minimal cost path in a cluttered environment with static or dynamic obstacles. For static environments, the shortest path problem can be formulated into a standard optimal control problem in which

$$\begin{aligned} &\text{(Dynamical System)} && \dot{x} = u(t), \quad t_0 \le t \le t_f, \quad x(t) \in \mathbf{R}^n, \\ &\text{(Performance Index)} && J = \int_{t_0}^{t_f} |u(t)| \, \mathrm{d}t, \\ &\text{(Path Constraints)} && \phi(x(t)) \ge 0, \quad x(t_0) = X, \quad x(t_f) = Y, \end{aligned} \tag{3}$$

where X, Y are starting and ending points and $\phi(x)$ is the level set function of the

obstacles. And for dynamic environments, the minimal cost path is the minimizer of

$$\begin{aligned} &\text{(Dynamical System)} && \dot{x} = u(t), \quad t_0 \le t \le t_f, \quad x(t) \in \mathbf{R}^n, \\ &\text{(Performance Index)} && J = \int_{t_0}^{t_f} L(|u(t)|)\,\mathrm{d}t, \\ &\text{(Path Constraints)} && \phi(t, x(t)) \ge 0, \quad x(t_0) = X, \quad x(t_f) = Y, \end{aligned} \tag{4}$$

where L is the running cost and $\phi(t, x)$ is time-dependent level set functions representing the dynamic environment. Under certain conditions, we will show that the optimal paths of both (3) and (4) are separable. We will demonstrate the efficiency of the method of evolving junctions with several different settings:

1. Shortest path in a static environment in $\mathbf{R}^2$ or $\mathbf{R}^3$ where the obstacles have piecewise smooth boundaries;

2. Shortest path in a static environment in $\mathbf{R}^3$ where the obstacles are polyhedrons;

3. Shortest path in a static environment where the mover is a disk or the starting point and ending point are sets;

4. Shortest path in a dynamic environment where obstacles disappear or appear;

5. Shortest path in a dynamic environment where obstacles are moving.

In summary, the method has the following advantages comparing to the existing methods

1. Fast. Because only initial value stochastic differential equations need to be solved, it is fundamentally faster than solving PDEs, Boundary value ODEs and constrained NLPs.

2. Ability to find a globally optimal path as well as a series of locally optimal paths by the adoption of intermittent diffusion.

CHAPTER II

MATHEMATICAL PREPARATION

In this chapter, we briefly describe the mathematics background of some major components of the algorithm. Although all the examples we conduct will be in the Euclidean space $\mathbf{R}^3$, we will formulate the problem with maximal generality in metric spaces. The advantages of this formulation are three folds (1) The main component of the method is gradient descent which can be formulated in metric spaces. This allows future possible extensions; (2) It allows different geometries, whether endowed with a differential structure or not to be dealt with in a uniform manner; (3) It's evident that the problem is related to optimal transport which is built upon metric spaces.

2.1 Optimal Control

Optimal control theory, also known as trajectory optimization, is a subject where it seeks to determine the input (control) to a dynamical system that optimize a given performance functional (maximize profit, minimize cost, etc) while satisfying different kinds of constraints. It is closely related to this thesis in the following way. First, the problem itself can be entirely formulated in the framework of optimal control. In addition, the numerical methods for optimal control is highly mature and there exist many packages dealing with optimal control problem. However, due to some intrinsic drawback of those numerical methods, we will not approach the shortest path problem in this way. From this point view, one may expect that our approach works for general optimal control problems. Second, we will use optimal control theory as an analytical tool in Chapter 6, although this is viable in rare cases.

There are two different approaches to optimal control problem, one is based on

calculus of variation, also known as Pontryagin Maximum Principle, one is based on the principle optimality which states that each subarc of the optimal trajectory must also be optimal. We will describe them separately.

Consider the system described by the following differentia equations:

$$\dot{x} = f(x(t), u(t), t), \quad t_0 \leq t \leq t_f, \quad x(t) \in \mathbf{R}^n, \tag{5}$$

where $u(t) \in \mathbf{R}^m$ is the control. Associated with dynamical system is the following performance index (cost) of the form

$$J = \psi(x(t_f), t_f) + \int_{t_0}^{t_f} L(x(t), u(t), t)\, \mathrm{d}t, \tag{6}$$

where $L\colon \mathbf{R}^n \times \mathbf{R}^m \times \mathbf{R}^+ \to R$ is the running cost, also called the Lagrangian and $\phi\colon \mathbf{R}^n \times \mathbf{R}^+ \to \mathbf{R}$ is the terminal cost. t_f is the terminal time which may be undetermined. One can also impose various constraints on the path or control, for example, to require the final position $x(t_f)$ been fixed or the whole path outside certain region. Define a scalar function $H\colon \mathbf{R}^n \times \mathbf{R}^m \times \mathbf{R}^n \times \mathbf{R}^+ \to \mathbf{R}$, called the Hamiltonian as follows

$$H(x, u, \lambda, t) = L(x, u, t) + \lambda^T f(x, u, t), \tag{7}$$

where $\lambda \in \mathbf{R}^n$. Using calculus of variation, we obtain a control or path is a minimizer if

$$\begin{gathered}
\dot{x} = f(x, u, t), \\
\dot{\lambda} = -\left(\frac{\partial f}{\partial x}\right)\lambda - \left(\frac{\partial L}{\partial x}\right)^T, \\
H(x(t), u(t), \lambda(t), t) = \min H(x(t), v, \lambda(t), t) \quad \text{for any feasible } v.
\end{gathered}$$

The boundary conditions are spit in the sense that some are given for $t = t_0$ and some are given for $t = t_f$.

$$\begin{aligned}
x(t_0) &= x_0, \\
\lambda(t_f) &= \left(\frac{\partial \psi}{\partial x}\right)^T.
\end{aligned}$$

The resulted system is a two-point boundary value problem. We stress here that the above condition is only necessary and usually leads to local minimizing trajectories. Now we consider various extension of the standard formulation.

2.1.1 State Variable Specified at Fixed Terminal Time

If $x(t_f)$ is specified for the first p coordinate, i.e.

$$x(t_f) = (\overline{x}_1, \overline{x}_2, \cdots, \overline{x}_p, x_{p+1}, \cdots, x_n) \tag{8}$$

where $\overline{x}_k$ denotes specified values, then the boundary condition becomes

$$x_k(t_f) = \overline{x}_k, \quad k = 1, \cdots, p, \tag{9}$$

$$\lambda_k(t_f) = \frac{\partial \psi}{\partial x_k}(t_f), \quad k = p+1, \cdots, n. \tag{10}$$

The other values are left undetermined. There are still n terminal boundary conditions in total.

2.1.2 State Variable Specified at Unspecified Terminal Time

Now assume the terminal time is unspecified. This occurs, for example, when the performance index is simply the time. It is convenient to regard t_f as a control variable to be choose in addition to the control functions $u(t)$. The optimality condition in this case is

$$\dot{x} = f(x, u, t), \tag{11}$$

$$\dot{\lambda} = -\left(\frac{\partial f}{\partial x}\right)\lambda - \left(\frac{\partial L}{\partial x}\right)^T, \tag{12}$$

$$H(x(t), u(t), \lambda(t), t) = \min H(x(t), v, \lambda(t), t) \quad \text{for any feasible } v, \tag{13}$$

$$(\psi_t + H)(t_f) = 0 \tag{14}$$

with boundary condition

$$x_k(t_f) = \overline{x}_k, \quad k = 1, \cdots, p, \tag{15}$$

$$\lambda_k(t_f) = \frac{\partial \psi}{\partial x_k}(t_f), \quad k = p+1, \cdots, n, \tag{16}$$

if $x(t_f)$ is specified as $x(t_f) = (\overline{x}_1, \overline{x}_2, \cdots, \overline{x}_p, x_{p+1}, \cdots, x_n)$.

2.1.3 Optimal Control with Path Constraint

To formulate our problem in terms of optimal control, we need to impose additional constraint on the path, i.e. to require the path to avoid all the obstacles. Mathematically, we consider

$$\dot{x} = f(x(t), u(t), t), \quad t_0 \leq t \leq t_f, \quad x(t) \in \mathbf{R}^n \tag{17}$$

with constraint

$$x(t_0) = x_0, \tag{18}$$

$$\phi(x, u, t) \geq 0, \tag{19}$$

$$x(t_f) = x_f. \tag{20}$$

ϕ: $\mathbf{R}^n \to \mathbf{R}$ can be regarded as the level set of the obstacle. The dependence of ϕ explicitly on time t occurs , for example, when considering moving obstacles.

The problem can be handled by adding an extra term to the Hamiltonian, i.e. define

$$H = L + \lambda^T f + \mu\phi, \tag{21}$$

where

$$\mu \begin{cases} > 0, & \phi = 0; \\ = 0, & \phi > 0 \end{cases} \tag{22}$$

and the Euler-Lagrange equation becomes

$$\dot{\lambda}^T = -\frac{\partial H}{\partial x} = \begin{cases} -L_x - \lambda^T f_x - \mu\phi_x, & \phi = 0; \\ -L_x - \lambda^T f_x, & \phi > 0. \end{cases} \tag{23}$$

All the rest procedures are exactly the same as the control problem without path constraint, for example, the optimal control is determined by $H_u = 0$.

If the path constraint doesn't explicitly depends on control u, i.e.

$$\phi(x, u, t) = \phi(x, t), \tag{24}$$

one take successive total time derivatives of $\phi(x, t)$ and substitute $f(x, u, t)$ for $\dot{x}$ until the expression is explicitly dependent on u. Denote p the number of derivatives required. $\phi^{(p)}$ depends on u and we can apply the above approach, namely, by choosing the Hamiltonian as

$$H = L + \lambda^T f + \mu\phi^{(p)}. \tag{25}$$

For an illustration of the above technique, let's consider the following simple version of our problem

$$\dot{x} = u, \quad x \in \mathbf{R}^2, \quad u \in \mathbf{R}^2, \tag{26}$$

$$x(0) = X, \quad x(t_f) = Y, \tag{27}$$

$$\phi(x(t)) \geq 0 \tag{28}$$

with performance index the length of the path

$$J = \int_0^{t_f} |u| \, \mathrm{d}t. \tag{29}$$

Since $\dot{\phi} = \nabla\phi(x) \cdot \dot{x} = \nabla\phi(x) \cdot u$, the new Hamiltonian is

$$H = |u| + \lambda^T u + \mu\nabla\phi(x) \cdot u. \tag{30}$$

2.1.4 Dynamical Programming

Dynamical programming is yet another approach to optimal control. Contrary to the approach based on Pontryagin maximum principle which gives rise to a two-boundary value problem, dynamical programming approach results in a partial differential equation, called Hamilton-Jacobi-Bellman equation. Another notable feature of dynamical programming is that the solution to the HJB equation is the global minimizer of the

control problem. We will give a very informal derivation here. Consider optimal control problem (5) and define

$$J(x,t) = \min_u \psi(x(t_f), t_f) + \int_t^{t_f} L(x(s), u(s), s)\,\mathrm{d}s, \tag{31}$$

where $x(t) = x$ is the initial condition The boundary condition of $J(x,t)$ depends on the boundary condition of the original problem. For example, if the state variable is specified at the terminal time $x(t_f) = \overline{x}$ then the boundary condition of $J(x,t)$ is

$$J(x,t_f) = \overline{x}. \tag{32}$$

Marching forward by time Δt, $x(t)$ moves to

$$x(t+\Delta t) = f(x,u,t)\Delta t. \tag{33}$$

Therefore,

$$\begin{aligned} J(x,t) &= \min_u \left\{ J(x + f(x,u,t)\Delta t, t+\Delta t) + \int_t^{t+\Delta t} L(x,u,s)\,\mathrm{d}s \right\} \\ &= \min_u \left\{ J(x + f(x,u,t)\Delta t, t+\Delta t) + L(x,u,t)\Delta t \right\} \\ &= \min_u \left\{ J(x,t) + \frac{\partial J}{\partial x} f(x,u,t)\Delta t + \frac{\partial J}{\partial t}\Delta t + L(x,u,t)\Delta t \right\}. \end{aligned}$$

Taking limit as $\Delta t \to 0$, we have

$$-\frac{\partial J}{\partial t} = \min_u \left\{ L(x,u,t) + \frac{\partial J}{\partial x} f(x,u,t) \right\}. \tag{34}$$

The minimization above is not necessarily over all possible $u \in \mathbf{R}^m$. Depending on different scenarios, the feasible set of control vary accordingly. For example, consider problem (5) with path constraint

$$\phi(x) \geq 0, \tag{35}$$

the feasible set of optimal control $F(x)$ is

$$F(x) = \begin{cases} \mathbf{R}^m, & \phi(x) > 0; \\ \{\, u \mid \nabla\phi(x)\cdot u \geq 0 \,\}, & \phi(x) = 0. \end{cases} \tag{36}$$

The second case is because the direction must point outside of the obstacle, otherwise the object will enter the obstacle. If the obstacles are moving, i.e. the path constraint is

$$\phi(x,t) \geq 0. \tag{37}$$

Then when $\phi(x,t) = 0$, the feasible control u must satisfy

$$\nabla\phi(x,t) \cdot u + \phi_t(x,t) \geq 0. \tag{38}$$

2.2 Shortest Path in Metric Space

2.2.1 Length space

Let (X,d) be a metric space with metric $d\colon X \times X \to \mathbf{R}^+$. The metric d satisfies the following properties:

$$d(x,y) \geq 0 \text{ and } d(x,y) = 0 \Leftrightarrow x = y, \tag{39}$$

$$d(x,y) = d(y,x), \tag{40}$$

$$d(x,y) \leq d(x,z) + d(z,y). \tag{41}$$

A path in a metric space is a continuous map $\gamma\colon I \to X$ defined on a closed interval $I \subset \mathbf{R}$. $t \in I$ usually denotes time, but not necessarily. A length space is the triple $(X, \mathcal{A}, L)$ where X is the underlying metric space, $\mathcal{A}$ is the set of admissible paths and L is the length of paths in $\mathcal{A}$. The set of admissible paths has to satisfy the following assumptions:

1. $\mathcal{A}$ is closed under restriction. If $\gamma\colon [s,t] \to X$ is an admissible path and $s \leq q \leq r \leq t$, the the restriction $\gamma\colon [q,r] \to X$ is also admissible.

2. $\mathcal{A}$ is closed under concatenations. That is if $\gamma_1\colon [r,s] \to X$ and $\gamma_2\colon [s,t] \to X$ are both admissible and $\gamma_2(s) = \gamma_1(s)$, then the path $\gamma\colon [r,t] \to X$ defined by

$$\gamma(\theta) = \begin{cases} \gamma_1(\theta), & \theta \in [r,s]; \\ \gamma_2(\theta), & \theta \in [s,t] \end{cases}$$

is also admissible. We denote $\gamma = \gamma_1 \cdot \gamma_2$.

3. $\mathcal{A}$ is closed under linear reparameterization. Namely, for an admissible path $\gamma\colon [s,t] \to X$ and function $\phi\colon [q,r] \to [s,t]$ of the form $\phi(t) = at + b$, $\gamma \circ \phi(t)$ is also admissible path.

One thing to note here is that the requirement of closeness under reparemeterization prevents the interpretation of the parameter as time variable. This will be loosed when considering optimal path in dynamic environments where the parameter is required to be time variable.

The length functional $L\colon \mathcal{A} \to \mathbf{R}^+$ has to satisfy the following properties

1. Length of paths is additive, i.e. $L(\gamma_1 \cdot \gamma_2) = L(\gamma_1) + L(\gamma_2)$.

2. L depends continuously on the length of the domain I. In other words, if one define $L(\gamma, s, \theta) = L(\gamma|_{s,\theta}), s, \theta \in I$, then $L(\gamma, a, \theta)$ as a function of θ is continuous.

3. The length is invariant under reparameterizations, i.e. $L(\gamma \circ \phi) = L(\gamma)$.

Again, when dealing with optimal path in dynamic environment, property (3) will be dropped. The simple reason is that time variable can not be reversed, which is equivalent to reparameterization by $\phi(t) = -t + b$.

The length structure induces a new metric in X in the following way. Denote $\mathcal{A}(x,y)$ all the admissible paths that starts from x and ends at y,

$$\mathcal{A}(x,y) = \{\gamma\colon [s,t] \to X \in \mathcal{A} \mid \gamma(s) = x, \gamma(t) = y\}. \tag{42}$$

Then the new metric $d_L(x,y)$ is the minimal lengths of all the paths connecting x and y,

$$d_L(x,y) = \inf_{\gamma \in \mathcal{A}(x,y)} L(\gamma). \tag{43}$$

Definition 1 *A metric that can be obtained as the distance function associated to a length structure is called a intrinsic metric. A metric space whose metric is intrinsic is called a length space.*

It is easy to see $\mathbf{R}^n$ with metric $d(x,y) = \|x-y\|$ is a length space. There exists, of course, metric space that is not intrinsic. For example, the union of segments

$$\bigcup_{i=1}^{\infty}[(0,0),(\cos\frac{1}{i},\sin\frac{1}{i})] \cup [(0,0),(1,0)],$$

which is a subset $\mathbf{R}^2$ is not a length space. The point is that a length space must be locally path connected, i.e. every neighborhood of any point contains a smaller neighborhood which is path connected.

A natural way that a length structure can arise is to induce by the metric. Under the new induced metric, any metric space becomes a length space.

Definition 2 *Let (X,d) be a metric space and $\gamma\colon [s,t] \to X$ be a path in X. A partition π of $[s,t]$ is a finite collection of points $\pi = \{\,\theta_0,\cdots,\theta_n\,\}$ such that $s = \theta_0 < \theta_1 < \cdots < \theta_n = t$. The length of γ over partition π is defined to be*

$$L_\pi(\gamma) = \sum_{i=0}^{n-1} d(\gamma(\theta_i),\gamma(\theta_{i+1})).$$

Now one can define the length of γ to be the supremum of $L_\pi(\gamma)$ over all partitions

$$L_d(\gamma) = \sup_\pi L_\pi(\gamma).$$

A curve with finite induced metric is called rectifiable. For a metric space (X,d), let $\mathcal{A}$ be the set of admissible paths consisting all the continuous paths. Then $(X,\mathcal{A},L_d)$ is a length space

$$(X,d) \to (X,\mathcal{A},L_d).$$

The induced length space in turn induces a new metric in X, one has

$$(X,d) \to (X,\mathcal{A},L_d) \to (X,d_{L_d}).$$

2.2.2 Shortest Path

When speaking of a path, it is important to distinguish its map and its image. Throughout the thesis, we will use path to refer the continuous function and curve the image of the function. Different maps can lead to identical curves. In other words, if there exists a nondecreasing continuous function $\phi\colon I_1 \to I_2$ such that $\gamma_1 = \gamma_2 \circ \phi$, then γ_1 and γ_2 have the same image. We say curve γ_1 and γ_2 are the same

Definition 3 *A path $\gamma\colon I \to X$ is said to be arc-length parameterized if $L(\gamma, \theta_1, \theta_2) = |\theta_2 - \theta_1|$.*

Theorem 4 *Every rectifiable curve γ admits a arc-length parameterization.*

Definition 5 *A curve $\gamma^*\colon [s,t] \to X$ is a shortest path connecting x and y if its length is minimal among all the curves with the same endpoints. Mathematically,*

$$\gamma^* = \operatorname{argmin}_{\gamma \in \mathcal{A}(x,y)} L(\gamma).$$

The example of $\mathbf{R}^2 \setminus \{0\}$ shows that the shortest path may not necessarily exist. We will state the following existence theorem without proof.

Theorem 6 *If (X, d) is a complete locally compact length space, then for any $x, y \in X$ such that $d(x, y) < \infty$ there exists a shortest path connecting x and y.*

2.3 Intermittent Diffusion

2.3.1 Algorithm

In this section, intermittent diffusion, a main component of our algorithm, is introduced. We will abbreviate intermittent diffusion by ID hereafter. Intermittent Diffusion is a global optimization strategy developed in [16]. It concerns the following minimization problem

$$\min_x g(x), \quad x \in \Omega, \tag{44}$$

where $\Omega \in \mathbf{R}^n$ is the admissible set and $g(x)$ is the objective functional, usually defined by an energy functional or a cost functional.

The main idea of finding the global minimizer is to introduce randomness into the process in some way. For example, Monte-Carlo sampling, genetic algorithm, Metropolis random walk, simulated annealing all fall into this category [35, 45, 68]. By combining the idea of simulated annealing and gradient descent, [11, 1] proposed to solve the problem by running the following stochastic differential equation

$$dx(t,\omega) = -\nabla g(x(t,\omega))dt + \sigma(t)dW(t), \quad t \in [0,\infty), \tag{45}$$

where $W(t)$ is the standard Brownian motion in $\mathbf{R}^n$, and $\sigma(t)$ corresponds to the cooling rate in simulated annealing. It can be shown that if the cooling rate decrease with certain rate, i.e.

$$\sigma(t) = \frac{c}{\sqrt{\log t}}, \tag{46}$$

the solution of (45) converges weakly to a distribution concentrated at the global minima of g.

ID use the same formulation as (45) but choose a more clever $\sigma(t)$, i.e. a piecewise constant function

$$\sigma(x,t) = \sum_{j=1}^{N} \sigma_j \chi_{[S_j,T_j]}(t), \tag{47}$$

where $0 = S_1 < T_1 < S_2 < T_2 < \cdots < S_N < T_N < S_{N+1} = T$ and $\chi_{[S_j,T_j]}$ is the characteristic function of interval $[S_j, T_j]$. The rationale behind this particular choice is the following. First, when $\sigma(t) = 0$, the flow becomes a gradient descent flow which converges to a local minimizer; when $\sigma(t) > 0$, the trajectory of (45) have a certain probability controlled by $\sigma(t)$ of jumping out the local trap. Second, this particular choice actually mimic the process of annealing better that (45). In reality, the annealing process often consists of a series of alternating heating and cooling subprocesses which is exactly what (47) manifests.

We will state the algorithm first, followed by a proof of the convergence. The reason we include the proof here is to analyze the converge rate of the algorithm.

Algorithm: Intermittent Diffusion

Input: functional to optimize,

number of intermittent diffusion intervals N.

maximal diffusion time $T_{\max}$ and maximal diffusion strength $\sigma_{\max}$.

Output: A collection of local minimizers including the global one.

Initialization. Generate a random initial state $x_0 \in \Omega$, $x_{opt} = x_0$

for $l = 1 : N$

Uniformly sample $\sigma \in [0, \sigma_{\max}]$ and $T \in [0, T_{\max}]$

Compute the stochastic differential equation for $t \in [0, T]$

$$dx(t, \omega) = -\nabla g(x(t, \omega))dt + \sigma dW(t), \quad x(0) = x_{opt}$$

and record the final state $x_T = x(T, \omega)$

Compute the solution to the following gradient flow until convergence

$$\dot{x}(t) = -\nabla g(x(t)), \quad x(0) = x_T$$

and record the final state x_l. If $g(x_l) < g(x_{opt})$, $x_{opt} = x_i$

end

Return $x_1, \cdots, x_N$ as N local minimizers and x_{opt} the global minimizer.

Remark 1 *1. Step 5 can be solved by many well established schemes, for example, Euler scheme and Runge-Kutta scheme.*

2. The stop criterion in Step 5 can be set in different ways. For simplicity, in our implementation, the iteration stops if the absolute value of the difference

between two successive iterates is less that a prescribed tolerance error ϵ.

3. *The stochastic differential equation in Step 4 can be discretized as follows*

$$x_{n+1} - x_n = -\nabla g(x_n)\Delta t + \sigma\sqrt{\Delta t}\xi, \quad x_0 = x_{opt}, \tag{48}$$

where Δt is the step size and ξ is a random variable following standard normal distribution.

4. *The algorithms works equally well for functional on Riemannian manifold and metric space where the gradient flow and brownian motion are well defined. For a detailed description, see 2.4*

5. *The inputs usually $\sigma_{\max}$ depends on the scale of the problem while the diffusion time depends on the error tolerance. See 2.3.3 for the discussion.*

2.3.2 Proof of Convergence

We will restate the proof in [16] here in order to show the convergence rate.

Theorem 7 *Let Q be the set of global minimizers, U a small neighborhood of Q and X_{opt} the optimal solution obtained by ID. Then for any given $\epsilon > 0$, there exist $\tau > 0, \sigma_0 > 0, N_0 > 0$ such that if $T_i - S_i > \tau, \sigma_i < \sigma_0, i = 1, \cdots, N$ and $N > N_0$,*

$$P(X_{opt} \in U) \geq 1 - \epsilon. \tag{49}$$

Proof 1 *For simplicity, assume that there is only one global minimizer x^*. We refer readers to [16] for a general treatment. Because the Hessian matrix at x^* is negative definite, we can always take $U = B(x^*, r)$, a ball centered at x^* with sufficiently small radius r.*

Consider one segment for $t \in [S_i, S_{i+1}]$. Since the trajectory rests on a small neighborhood of a local minimizer q_i at the end of each gradient descent period, we may assume $x(S_i) \in B(q_i, r_i)$. For $t \in [S_i, T_i]$, ID is a diffusion process. Denote

the probability density function of $x(t)$ visiting x at time $t \in [S_i, T_i]$ by $p_{\sigma_i}(t - S_i, x)$. We make the dependence on σ_i explicitly here. It is well known that p_{σ_i} satisfies the Fokker-Planck equation given by

$$(p_{\sigma_i})_t = \nabla \cdot (\nabla g(x) p_{\sigma_i}) + \frac{1}{2}\sigma_i^2 \Delta p_{\sigma_i}. \tag{50}$$

As a solution of this linear convection diffusion equation, $p_{\sigma_i} \in C^\infty$ continuously depends on t, σ_i.

It is straightforward to verify that [58] has an equilibrium solution which is given by the famous Gibbs distribution

$$\bar{p}_{\sigma_i}(x) = A e^{-\frac{2g(x)}{\sigma_i^2}}, \tag{51}$$

where A is the normalizing constant defined by

$$A = \left(\int_{\mathbf{R}^n} e^{-\frac{2g(x)}{\sigma_i^2}} \, \mathrm{d}x \right)^{-1}. \tag{52}$$

According to [11] and the references cited therein, $\bar{p}_{\sigma_i}(x)$ weakly converges to a point distribution concentrated at x^* as $\sigma_i \to 0$. This implies that

$$\lim_{\sigma_i \to 0} \int_U \bar{p}_{\sigma_i} \, \mathrm{d}x = 1. \tag{53}$$

Now define the attraction set of q_i as

$$K(q_i) = \{x \mid \textit{gradient flow } (45) \textit{ with } \sigma = 0 \textit{ and initial state } x \textit{ converges to a point in } B(q_i, r)\}.$$

By the definition of $B(q_i, r)$, we have $B(q_i, r) \subset K(q_i)$ provided r is sufficiently small. Thus for any given $\alpha \in (0, 1)$, there exists σ_0 such that for $\sigma_i < \sigma_0$,

$$\int_U \bar{p}_{\sigma_i}(x) \, \mathrm{d}x > \alpha + \frac{1 - \alpha}{2}. \tag{54}$$

Also there exists a $\tau(\alpha)$ such that for $T_i - S_i > \tau(\alpha)$,

$$\| p_{\sigma_i}(T_i - S_i) - \bar{p}_{\sigma_i}(x) \|_{L^1} < \frac{1 - \alpha}{2}. \tag{55}$$

Combining (166),(55), *we have*

$$\int_U p_{\sigma_i}(T_i - S_i, x)\,\mathrm{d}x > \alpha. \tag{56}$$

This implies that

$$P(x(S_{i+1}) \in U \mid x(S_i) \in B(q_i, r_i)) = P(x(T_i) \in K(x^*) \mid x(S_i) \in B(q_i, r_i)) > \alpha \tag{57}$$

due to the fact that ID is deterministic on $[T_i, S_{i+1}]$. *Let* Θ *be the set of all local minimizers of* g, *then*

$$\begin{aligned} P(x(S_{i+1}) \in U^c) &= \sum_{q_i \in \Theta} P(x(S_{i+1}) \in U^c \mid x(S_i) \in B(q_i, r_i)) P(x(S_i) \in B(q_i, r_i)) \\ &< (1-\alpha) \sum_{q_i \in \Theta} P(x(S_i) \in B(q_i, r_i)) \\ &< 1 - \alpha. \end{aligned}$$

The above estimate can be made for all segments $i = 1, 2, \cdots, N$ *uniformly. Therefore,*

$$P(X_{opt} \in U^c) = P(\bigcap_{i=1}^{N} \{x_{S_i} \in U^c\}) < (1-\alpha)^N. \tag{58}$$

Thus for any given $\epsilon \in (0,1)$, *there exists* $N_0 > 0$, *such that as long as* $N > N_0$,

$$P(X_{opt} \in U^c) < (1-\alpha)^N < \epsilon. \tag{59}$$

2.3.3 Convergence Analysis

From the proof, it's easy to derive the time that required to reach the global minimizer. Let $T_1 = \tau(\alpha)$ be the time required for the corresponding Fokker-Planck equation to be within a distance $(1-\alpha)/2$ from the equilibrium and T_2 be the time needed for the gradient flow to converge. Under the assumption that every minimizer of $g(x)$ is hyperbolic,

$$T_2(\epsilon) = \mathbf{O}(\log \frac{1}{\epsilon}). \tag{60}$$

The analysis of T_2 boils down to analyzing the convergence rate of the underlying Fokker-Planck equation. This has been intensively studied in the past two decades

[43, 3, 31]. We give a brief introduction here following [43]. Consider the initial value problem for the Fokker-Planck equation

$$\rho_t(t,x) = \nabla \cdot (D(\nabla\rho + \rho\nabla V)), \quad t > 0, \quad x \in \mathbf{R}^n, \tag{61}$$

$$\rho(0,x) = \rho_0, \tag{62}$$

where $D(x)$ is symmetric, locally uniformly positive definite matrix, and $V(x)$ a confinement potential. Assume that the initial value is a density function, i.e.

$$\int \rho_0 \,\mathrm{d}x = 1, \tag{63}$$

then one easily verifies that (61) has steady state

$$\rho_\infty = e^{-V(x)} \tag{64}$$

assuming that V to be normalized as $\int e^{-V(x)}\,\mathrm{d}x = 1$. For any two density functions ρ_1, ρ_2, define the relative entropy of ρ_1 with respect to ρ_2 by

$$H(\rho_1|\rho_2) = \int_{\mathbf{R}^n} \log\frac{\rho_1}{\rho_2}\rho_2(\,\mathrm{d}x). \tag{65}$$

In particular, one can consider the relative entropy of $\rho(t)$ with respect to ρ_∞, i.e.

$$H(\rho(t)|\rho_\infty), \tag{66}$$

as well as its derivative

$$I(\rho(t)|\rho_\infty) = \frac{d}{dt}H(\rho(t)|\rho_\infty). \tag{67}$$

I is known in information theory as the relative information. It can be shown that the relative entropy and relative information both decay exponentially.

Theorem 8 *Assume $H(\rho(t)|\rho_\infty) < \infty$ and the scalar coefficients $V(x), D(x)$ satisfies the following condition: there exists $\lambda > 0$ such that*

$$(\frac{1}{2} - \frac{n}{4})\frac{1}{D}\nabla D \otimes \nabla D + \frac{1}{2}(\Delta D - \nabla D \cdot \nabla V)I + D\frac{\partial^2 V}{\partial x^2} + \frac{1}{2}(\nabla V \otimes \nabla D + \nabla D \otimes \nabla V) - \frac{\partial^2 D}{\partial x^2} \geq \lambda I. \tag{68}$$

Then the relative information and relative entropy converge to 0 exponentially

$$|I(\rho_t|\rho_\infty)| \leq e^{-2\lambda t}|I(\rho_0|\rho_\infty)|, \tag{69}$$

$$H(\rho_t|\rho_\infty) \leq e^{-2\lambda t}H(\rho_0|\rho_\infty). \tag{70}$$

Remark 2 *For $D(x) = I$ condition* (68) *reduces to: there exists λ such that*

$$\frac{\partial^2 V(x)}{\partial^2 x} \geq \lambda I. \tag{71}$$

However, this is less interesting since (71) *implies that the functional $V(x)$ is convex. Therefore there is a unique local minimizer. In order to deal with nonconvex case, we have the Holley-Stroock perturbation lemma [31]:*

Theorem 9 *If V can be written as $V = V_0 + v$, where $v \in L^\infty$ and e^{-V_0} satisfies a logarithmic Sobolev inequality with constant λ, then also e^{-V} satisfies a logarithmic Sobolev inequality with constant $\lambda e^{-osc(v)}$ with $osc(v) = \sup v - \inf v$.*

With the help of above theorem and Csiszar-Kullback inequality which states

$$\|\rho_1 - \rho_2\|_1^2 \leq 2H(\rho_1|\rho_2), \tag{72}$$

we have

$$2e^{-2\lambda t}H(p_0^i|\bar{p}_{\sigma_i}) < (\frac{1-\alpha}{2})^2. \tag{73}$$

Therefore, we can choose ($H = H(p_0^i|\bar{p}_{\sigma_i})$)

$$\tau(\alpha) = \frac{1}{\lambda}(\log\frac{2}{1-\alpha} + \frac{1}{2}\log 2H). \tag{74}$$

Also since $(1-\alpha)^N < \delta$, we choose

$$N = \frac{\log\delta}{\log(1-\alpha)}. \tag{75}$$

Thus the overall time needed is

$$N(T_1 + T_2(\epsilon)) = N(c_1 \log \frac{1}{\epsilon} + \frac{1}{\lambda} \log 2 + \frac{1}{\lambda N} \log \frac{1}{\delta} + \frac{1}{2\lambda} \log 2H) \tag{76}$$

$$= \frac{1}{\lambda} \log \frac{1}{\delta} + c_1 N \log \frac{1}{\epsilon} + N(\frac{1}{2\lambda} \log 8H) \tag{77}$$

$$= \frac{1}{\lambda} \log \frac{1}{\delta} + c_1 c_2 \log \frac{1}{\delta} \log \frac{1}{\epsilon} + c_2 \frac{1}{2\lambda} \log 8H \log \frac{1}{\delta} \tag{78}$$

$$= O(\log \frac{1}{\delta} \log \frac{1}{\epsilon}). \tag{79}$$

2.4 Gradient Flow in Metric Space

Gradient flows can be generalized to spaces which are not necessarily endowed with a differential structure. We will give a gentle introduction here to simplify our further discussion. The discussion basically follows [2].

Definition 10 *Let (X, d) be a complete metric space and let $v\colon (a, b) \to X$ be a curve. v is said to be absolutely continuous if there exists $m \in L^1(a, b)$ such that*

$$d(v(s), v(t)) \leq \int_s^t m(r)\, \mathrm{d}r, \quad \forall a < s \leq t < b. \tag{80}$$

Denote the set of absolutely continuous curves by $AC(a, b; X)$.

For any absolutely continuous curve, the scalar function

$$|v'|(t) = \lim_{s \to t} \frac{d(v(s), v(t))}{|s - t|} \tag{81}$$

exists almost everywhere for $t \in (a, b)$. $|v'(t)|$ can be regarded as the speed of the curve.

Let $\phi\colon X \to (-\infty, +\infty]$ be an extended real functional with proper domain

$$D(\phi) = \{\, v \in X \mid \phi(v) < \infty \,\} \neq \emptyset. \tag{82}$$

We would like to define the notion of "upper gradient" for ϕ.

Definition 11 *A function $g\colon X \to [0, +\infty]$ is a strong upper gradient for ϕ if for every absolutely continuous curve v, the function $g(v(t))$ is Borel and*

$$|\phi(v(t)) - \phi(v(s))| \leq \int_s^t g(v(r))|v'(r)|\,\mathrm{d}r, \quad \forall a < s \leq t < b. \tag{83}$$

A weaker notion can also be formulated, based on a point-wise formulation:

Definition 12 *A function $g\colon X \to [0, +\infty]$ is a weak upper gradient for ϕ if for every absolutely continuous curve v satisfying*

1. *$g(v)|v'| \in L^1(a, b)$;*

2. *$\phi(v)$ is equal to a function ϕ with finite point-wise variation a.e in (a, b),*

we have

$$|\phi'(t)| \leq g(v(t))|v'(t)|, \quad \text{for a.e.} t \in (a, b). \tag{84}$$

Now we are ready to state the definition of gradient. Instead defining the gradient as a functional, we consider the curves of maximal slope, which corresponds to the curve following the gradient flow.

Definition 13 *A locally absolutely continuous map $u\colon (a, b) \to X$ is a curve of maximal slope for the functional ϕ with respect to its upper gradient g, if $\phi \circ u$ is a.e. equal to a non-increasing map ϕ and*

$$\phi'(t) \leq -\frac{1}{2}|u'|^2(t) - \frac{1}{2}g^2(u(t)), \quad \text{for a.e. } t \in (a, b). \tag{85}$$

CHAPTER III

METHOD OF EVOLVING JUNCTIONS

3.1 Separability and Dimension Reduction

Consider the following optimal control problem with inequality path constraints. Let $x(t)$ and the control $u(t)$ be described by differential equations:

$$\dot{x}(t) = f(t, x(t), u(t)), \quad t \in [t_0, t_f], \quad x(t) \in \mathbf{R}^n, \quad u(t) \in \mathbf{R}^m, \tag{86}$$

with path constraints

$$x(t_0) = x_0, \quad M(t_f, x(t_f)) = 0, \quad \phi(t, x(t)) \geq 0, \quad t \in [t_0, t_f]. \tag{87}$$

The cost of the system is given by

$$J(u) = \psi(x(t_f), t_f) + \int_{t_0}^{t_f} L(t, x(t), u(t))\, \mathrm{d}t. \tag{88}$$

The optimal control problem is to find the control that minimizes the cost, i.e., to find

$$u^*(t) = \mathrm{argmin}_u\, J(u). \tag{89}$$

Definition 14 *A path $x(t)$ is said to be separable if there exists $t_0 < t_1 < t_2 < \cdots < t_n < t_{n+1} = t_f$ such that $x\,|_{[t_i, t_{i+1}]}$ are alternatively in the free space and on the boundary of the constraints ϕ. In other words,*

$$\phi(\theta, x(\theta)) \begin{cases} = 0, & \theta \in [t_i, t_{i+1}], \quad i \text{ odd;} \\ > 0, & \theta \in (t_i, t_{i+1}), \quad i \text{ even.} \end{cases} \tag{90}$$

The notion of separability is borrowed from [65] in which only paths consisting three parts are considered. We will denote $\tilde{x}_i = (t_i, x(t_i))$ and call them junctions. For any

two junctions $\tilde{x} = (s, x)$, $\tilde{y} = (t, y)$ and $\theta \in [s, t]$ define the optimal control in the free space as

$$J_0(\tilde{x}, \tilde{y}) = \min \int_s^t L(\theta, x(\theta), u(\theta))\, \mathrm{d}\theta,$$
$$\gamma_0(\tilde{x}, \tilde{y}) = \operatorname{argmin} \int_s^t L(\theta, x(\theta), u(\theta))\, \mathrm{d}\theta,$$

where

$$\dot{x}(\theta) = f(\theta, x, u), \quad x(s) = x, \quad x(t) = y, \quad \phi(\theta, x(\theta)) > 0, \quad \theta \in (s, t),$$

and the optimal control problem on the boundary of the constraints as (subscript "c" means constrained)

$$J_c(\tilde{x}, \tilde{y}) = \min \int_s^t L(\theta, x(\theta), u(\theta))\, \mathrm{d}\theta,$$
$$\gamma_c(\tilde{x}, \tilde{y}) = \operatorname{argmin} \int_s^t L(\theta, x(\theta), u(\theta))\, \mathrm{d}\theta,$$

where

$$\dot{x}(\theta) = f(\theta, x, u), \quad x(s) = x, \quad x(t) = y, \quad \phi(\theta, x(\theta)) = 0, \quad \theta \in [s, t].$$

The separability of the optimal path enables us to restrict our search of optimal paths in a subset H of the set all feasible paths. The set H is the collection of all paths that are determined by only the junctions on the boundaries the constraints. More precisely, for any path $\gamma \in H$, there exists a sequence of junctions on the boundary of the constraints $(\tilde{x}_0, \tilde{x}_1, \ldots, \tilde{x}_n, \tilde{x}_{n+1})$ of x with $\mathbf{x}_0 = (t_0, x_0)$, $M(t_n, x_{n+1}) = 0$ such that γ can be represented as

$$\gamma_0(\tilde{x}_0, \tilde{x}_1) \cdot \gamma_c(\tilde{x}_1, \tilde{x}_2) \cdot \gamma_0(\tilde{x}_2, \tilde{x}_3) \cdot \gamma_c(\tilde{x}_3, \tilde{x}_4) \cdots \gamma_0(\tilde{x}_n, \tilde{x}_{n+1}). \tag{91}$$

Here $\gamma_1 \cdot \gamma_2$ is the concatenation of two paths. As a result, if the optimal trajectory of (89) is separable, then the cost $J(\tilde{x}_0, \tilde{x}_{n+1})$ can be separated

$$J(\tilde{x}_0, \tilde{x}_1, \cdots, \tilde{x}_{n+1}) = \min \sum_{i=0}^{n} J(\tilde{x}_i, \tilde{x}_{i+1}). \tag{92}$$

Here

$$J(\tilde{x}_i, \tilde{x}_{i+1}) = \begin{cases} J_0(\tilde{x}_i, \tilde{x}_{i+1}), & i \text{ odd}; \\ J_c(\tilde{x}_i, \tilde{x}_{i+1}), & i \text{ even}. \end{cases}$$

Along with the finite dimensional problem (111), there is a hidden constraint. That is, the optimal trajectory in the free space connecting $\tilde{x}_{i-1}$ and $\tilde{x}_i$ must not intersect with the boundary of the constraints. Let's define the following visibility function for two junctions $\tilde{x} = (s, x)$ and $\tilde{y} = (t, y)$:

$$V(\tilde{x}, \tilde{y}) = \min_{s \le \theta \le t} \phi(\gamma_0(\tilde{x}, \tilde{y})(\theta)). \tag{93}$$

and the hidden constraint is

$$V(\tilde{x}_i, \tilde{x}_{i+1}) \ge 0, \quad \text{for } i \text{ even}. \tag{94}$$

As a result, in order to find the optimal trajectory, only the optimal junctions need to be computed. We gain a tremendous dimension reduction since the number of junctions is finite. In addition, in many situations, the subcontrol problem of computing J_0 and J_c is very easy. In other words, the optimal control problem (89) becomes

$$\min J(\tilde{x}_0, \tilde{x}_1, \cdots, \tilde{x}_{n+1}) \quad \text{s.t. } V(\tilde{x}_i, \tilde{x}_{i+1}) \ge 0, \quad \text{for } i \text{ even}. \tag{95}$$

3.2 Intermittent Diffusion with Constraints

To solve the above finite dimensional optimization problem, we will avoid using the standard nonlinear programming techniques, most of which are not capable of finding global minimizers. Instead, we will use first order intermittent diffusion in the following manner. Consider in general the following constrained optimization problem

$$\min f(x), \quad \text{s.t.} \quad C_I(x) \ge 0,$$

where $x \in \mathbf{R}^n$ and $C_I \colon \mathbf{R}^n \to \mathbf{R}^m$. Denote $A(x)$ the index of inequality constraints that are active at x. In other words,

$$A(x) = \{\, i \leq m \mid C_i(x) = 0 \,\} \tag{96}$$

and the feasible direction at x

$$\mathcal{F}(x) = \{\, p \in \mathbf{R}^n \mid \|p\| = 1, \nabla C_i(x) \cdot p \geq 0, \quad \text{for } i \in A(x) \,\}. \tag{97}$$

The negative gradient at x is the direction in which $f(x)$ decreases rapidly while maintain the feasibility. Hence the negative gradient direction can be defined as

$$-\frac{\nabla^c f(x)}{|\nabla^c f(x)|} = \text{argmin}_{p \in \mathcal{F}(x)} \nabla f(x) \cdot p, \tag{98}$$

$$|\nabla^c f(x)| = \min_{p \in \mathcal{F}(x)} |\nabla f(x) \cdot p|. \tag{99}$$

The constrained gradient can be computed by quadratic programming. More specifically, let $C_A = [\nabla C_{i1}, \nabla C_{i2}, \cdots, \nabla C_{ik}]^T, \quad i_j \in A(x)$ be the gradient of the active constraints. The negative gradient direction is the solution to the following optimization problem

$$\min \nabla f(x)^T p, \tag{100}$$

$$C_A p \geq 0, \tag{101}$$

$$p^T p \leq 1. \tag{102}$$

The Lagrangian is $L(p, \lambda, \mu) = \nabla f(x)^T p - \lambda^T C_A p - \mu(1 - p^T p)$ and the KKT conditions says

$$\nabla f(x) - C_A^T \lambda + 2\mu p = 0, \tag{103}$$

$$C_A p \geq 0, \tag{104}$$

$$p^T p \leq 1, \tag{105}$$

$$\lambda_i (C_A p)_i = 0. \tag{106}$$

The duality objective function is

$$q(\lambda,\mu)=\inf_{p} L(p,\lambda,\mu)=-\frac{1}{4\mu}\|\nabla f^T-\lambda^T C_A\|^2-\mu. \tag{107}$$

Hence the duality is

$$\max_{\lambda,\mu} -\frac{1}{4\mu}\|\nabla f^T-\lambda^T C_A\|^2-\mu, \quad \text{s.t } \lambda,\mu\geq 0, \tag{108}$$

which is equivalent to the following quadratic programming problem

$$\min_{\lambda} \lambda^T C_A C_A^T\lambda-2\lambda^T C_A\nabla f, \quad \lambda\geq 0. \tag{109}$$

Let λ_* be the minimizer of (109), then

$$\nabla^c f(x)=p=-(\nabla f-\lambda_*^T C_A). \tag{110}$$

Once the gradient is well defined, we can solve (95) by using Intermittent diffusion. Let $\tilde{x}=(\tilde{x}_0,\cdots,\tilde{x}_{n+1})$, then

$$\frac{d\tilde{x}}{d\theta}=-\nabla^c J(\tilde{x})+\sigma(\theta)dW(\theta). \tag{111}$$

Component-wisely, we hvae

$$\frac{d\tilde{x}_i}{d\theta}=-\nabla^c_{\tilde{x}_i}\left(J(\tilde{x}_{i-1},\tilde{x}_i)+J(\tilde{x}_i,\tilde{x}_{i+1})\right)+\sigma(\theta)dW(\theta). \tag{112}$$

3.3 Handling Dimension Changes

The benefit of using gradient descent is not only that it is simple and able to obtain the global minimizer when perturbation is introduced, but also that the visibility constraints (94) can be handled in a rather natural way. To maintain separability, we introduce the following two operations in the process

Insert junctions. During the evolution (111), $\gamma_0(\tilde{x}_k,\tilde{x}_{k+1})$ may intersect with the boundary of the constraints $\phi=0$. In other words, there may exist a θ such that

$$\phi(\theta,\gamma_0(\tilde{x}_k,\tilde{x}_{k+1})(\theta))\leq 0. \tag{113}$$

In order to maintain separability, we add the intersection points into the set of junctions. Let $(\tilde{x}_0, \tilde{x}_1, \cdots, \tilde{x}_n, \tilde{x}_{n+1})$ be the set of junctions representing the current path and assume $\gamma_0(\tilde{x}_k, \tilde{x}_{k+1})$ intersects with $\phi = 0$ at $\tilde{y}$ where $\phi(\tilde{y}) = 0$ for the first time (without loss of generality, assume there is only one such intersection). We add $\tilde{y}$ as a new junction and the path becomes

$$(\tilde{x}_0, \cdots, \tilde{x}_k, \tilde{y}, \tilde{y}, \tilde{x}_{k+1}, \cdots, \tilde{x}_n, \tilde{x}_{n+1}).$$

It is easy to see that the cost of the new path remains the same

$$\begin{aligned}
&J(\tilde{x}_0, \ldots, \tilde{x}_{n+1}) - J(\tilde{x}_0, \ldots, \tilde{x}_k, y, y, \tilde{x}_{k+1}, \cdots, \tilde{y}_{n+1}) \\
&\quad = J_0(\tilde{x}_k, \tilde{y}) + J_c(\tilde{y}, \tilde{y}) + J_0(\tilde{y}, \tilde{x}_{k+1}) - J_0(\tilde{x}_k, \tilde{x}_{k+1}) \\
&\quad = J_0(\tilde{x}_k, \tilde{y}) + J_0(\tilde{y}, \tilde{x}_{k+1}) - J_0(\tilde{x}_k, \tilde{x}_{k+1}) \\
&\quad = 0.
\end{aligned}$$

With the new set of junctions, we have another gradient flow for $\{\tilde{y}_k\}$ which is also generated by (111). However, the number of equations is strictly larger.

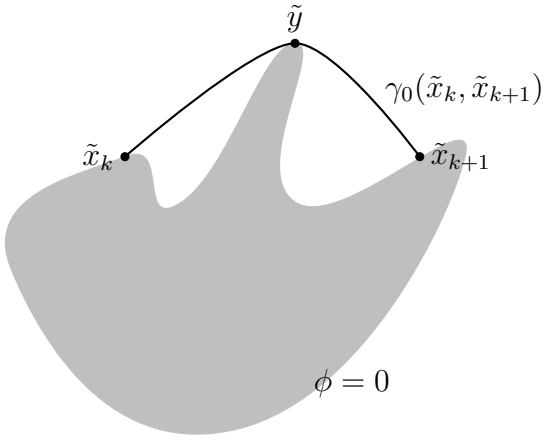


Figure 1: New junctions y will be added to the ODE system if $\gamma_0(\tilde{x}_k, \tilde{x}_{k+1})$ intersects with the boundary of the constraints.

Remove junctions. Junctions need to be removed if doing so results in a path with smaller cost. This case happens when two junctions $\tilde{x}_k$ and $\tilde{x}_{k+1}$ on the boundary

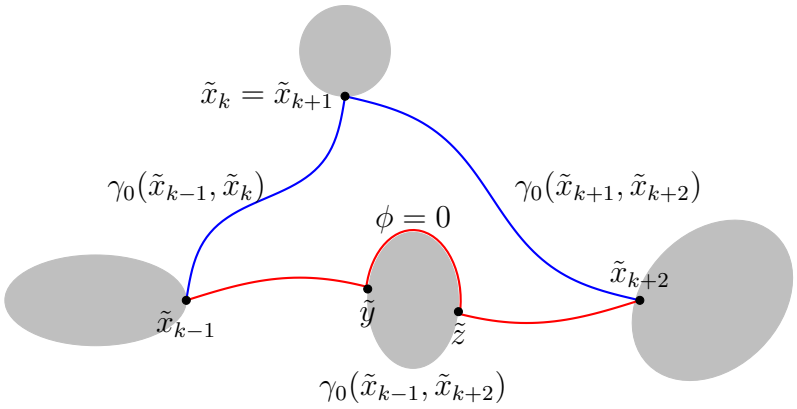


Figure 2: Remove junctions $\tilde{x}_k = \tilde{x}_{k+1}$ when they meet each other. The blue is the original path and the red is the new path.

meet each other during the flow, i.e. $\tilde{x}_k = \tilde{x}_{k+1}$. Since by triangle inequality, we have

$$J_0(\tilde{x}_{k-1}, \tilde{x}_{k+2}) \leq J_0(\tilde{x}_{k-1}, \tilde{x}_k) + J_0(\tilde{x}_{k+1}, \tilde{x}_{k+2}), \tag{114}$$

The original path $(\cdots, \tilde{x}_{k-1}, \tilde{x}_k, \tilde{x}_{k+1}, \tilde{x}_{k+2}, \cdots)$ can be shortened to obtained the path $(\cdots, \tilde{x}_{k-1}, \tilde{x}_{k+2}, \cdots)$. However, $\gamma_0(\tilde{x}_{k-1}, \tilde{x}_{k+2})$ may intersect with $\phi = 0$. Hence, to maintain separability, as in the insertion case, we add the intersections into the set of junctions. It should be noted that unlike the process of adding junctions, removing junctions causes a jump in the gradient flow. In addition, because of the possible adding process following the removing, the cost may even increase. For specific applications, this possible increasing of cost may be avoid by selecting a more dedicated the path touches the boundary.

3.4 Algorithm

With all the components discussed above, we are ready to state our algorithm.

Method of Evolving Junctions

Input: Constraint ϕ and M,

starting and ending points x and y,

Lagrangian L, terminal value ψ, and ODE f,

number of intermittent diffusion intervals m.

Output: The optimal set γ_{opt} of junctions.

---

1. Initialization. Find the initial path $\gamma^{(0)} = (\tilde{x}_0, \cdots, \tilde{x}_{n+1})$;
2. Select duration of diffusion $\Delta T_l, l \leq m$;
3. Select diffusion coefficients $\sigma_l, l \leq m$;
4. **for** $l = 1 : m$
5. $\quad \gamma^{(l)} = \gamma^{(0)}$;
6. $\quad$ **for** $j = 1 : \Delta T_l$
7. $\quad\quad$ Find $\nabla^c J(\gamma^{(l)})$ by solving quadratic programming (109);
8. $\quad\quad$ Update $\gamma^{(l)}$ according to (111) with $\sigma(t) = \sigma_l$;
9. $\quad\quad$ Remove junctions from or add junctions to $\gamma^{(l)}$ when necessary;
10. $\quad$ **end**
11. $\quad$ **while** $\|\nabla^c J(\gamma^{(l)})\| > \epsilon$
12. $\quad\quad$ Update $\gamma^{(l)}$ according to (111) with $\sigma(t) = 0$;
13. $\quad\quad$ Remove junctions from or add junctions to $\gamma^{(l)}$ when necessary;
14. $\quad$ **end**
15. **end**
16. Compare $J(\gamma^{(l)}), l \leq m$ and set $\gamma_{opt} = \text{argmin}_{l \leq m} J(\gamma^{(l)})$;

CHAPTER IV

SHORTEST PATH IN $\mathbf{R}^N$

4.1 Introduction

Finding the shortest path in the presence of obstacles is one of the fundamental problems in path planning and robotics. This problem can be described as follows:

Problem 1 *Given a finite number of obstacles in a region M in $\mathbf{R}^2$ or $\mathbf{R}^3$, what is the shortest path connecting two given points X, Y in M while avoiding the obstacles.*

Many techniques have been developed to tackle this problem. If the obstacles are polygonal, the problem can be reformulated as an optimization problem on a graph, and then solved by combinatoric methods. For example, in the shortest path map method, Hershberger and Suri [30] found an optimal $\mathbf{O}(n \log n)$ polynomial time algorithm where n is the total number of vertices of all polygonal obstacles in the plane $\mathbf{R}^2$. We refer to [38, 46] for a survey of the results and references therein.

However, Canny and Rief [7, 47] proved that this problem in $\mathbf{R}^3$ becomes NP-hard under the framework known as "configuration space". This is mainly because the shortest path doesn't necessarily pass through the set of vertices of polyhedrons. Instead, it may go through the interior points of edges, and this makes the optimal algorithm in 2-D fail. Two different approaches were developed later to overcome this difficulty. One is to find a path that is $1 + \epsilon$ times the length of the shortest one, for example, in [12, 52]. The idea is to subdivide the edges in certain ways and adopt the same optimal combinatorial methods which are effective in $\mathbf{R}^2$. However, The accuracy of the approximation algorithm is determined by the mesh size of the subdivision. In order to achieve a higher accuracy, one need to refine the mesh. Another approach, as studied in [32], is to build a 3D reduced visibility graph (RVG)

on the obstacles and extract a sequence of edges that contain the links of the shortest path. A energy is then defined in terms of those links which is then minimized by finding the critical points. The drawback of this approach is that the RVG only works for convex polyhedrons and they are not easy to compute. For similar works, we refer readers to [9, 10, 20].

When the obstacles are not polygonal, the combinatorial methods can not be applied directly. For this problem, commonly know methods are based on the theory of differential equations. For instance, in the planar path evolution approach, we can consider a one-parameter family of curves:

$$\gamma_t(\theta)\colon [0,1] \to \mathbf{R}^2$$

connecting $X = \gamma_t(0)$ and $Y = \gamma_t(1)$ defined by the following differential equation in t:

$$\frac{d\gamma_t(\theta)}{dt} = (\nabla W(\gamma_t(\theta)) \cdot \mathbf{n}(\theta))\mathbf{n}(\theta) - W(\gamma_t(\theta))\kappa(\theta)\mathbf{n}(\theta) \tag{115}$$

where $W\colon M \to \mathbf{R}$ is a penalty function, κ is the curvature of γ_t and $\mathbf{n}$ is the unit normal vector of γ_t. Equation (115) is derived by shrinking the length of the path γ_t while avoiding the obstacles. If a path intersects with an obstacle, then the penalty W imposed on the path would push the curve out and towards a locally optimal path. By choosing different penalty functions, this method also works for other path planning problems. We refer to [28, 69] for more discussions. However, since every point along the curve must be updated, this method may not be efficient, especially when the dimension of the problem is large. Moreover, it only leads to a locally optimal path. A different viewpoint of the path evolution method is to consider the steady state of (115) which satisfies

$$\nabla W(\gamma(\theta)) \cdot \mathbf{n} - W(\gamma(\theta))\kappa = 0, \quad \gamma(0) = X, \gamma(1) = Y. \tag{116}$$

This is a two point boundary value problem. Its numerical computation may become costly, especially in three or higher dimensions.

Another PDE-based approach is called front propagation. This approach is essentially the continuous version of Dijkstra method. The idea is to propagate a wave front from the starting point X with unit speed. The time the front first hits the ending point Y equals the length of the shortest path. It can be shown by the theory of optimal control that the arriving time T satisfies a PDE known as the Eikonal equation,

$$|\nabla T(x)|F(x) = 1, \quad T(X) = 0 \tag{117}$$

where $F(x)$ is the speed of the wave at point x. In this case, we have $F(x) = 1$. The Eikonal equation can be solved efficiently by fast marching method [63, 64] or fast sweeping [70, 74]. Similarly, by choosing a different speed function $F(x)$, front propagation can be extended to solve other path planning problems [59].

Moreover, there are also recent developments on the shortest path problem focusing on other aspects, for example, how to read the obstacles data. For interested readers, we refer to [62, 72, 22].

4.2 Formal Definition of the Shortest Path Problem

Problem 2 *Let (X, d) be a length space and $\{P_1, P_2, \cdots, P_N\}$ be N open subsets of X, representing obstacles, such that the boundary of each obstacle ∂P_k is also a complete locally compact length space. Given two points $x, y \in X_c = X \setminus \cup_{i=1}^{N} P_i$, define*

$$\mathcal{A}(x, y) = \{\, \gamma \colon [s, t] \to X_c \mid \gamma(s) = x, \gamma(t) = y \,\}.$$

Let $(X, \mathcal{A}, L)$ be the length structure induced by (X, d). Our problem is to find a shortest curve connecting x, y

$$\gamma^* = \operatorname{argmin}_{\gamma \in \mathcal{A}(x,y)} L(\gamma).$$

Let d_k be the intrinsic metric of ∂P_k and L_k the induced length structure on $(\partial P_k, L_k)$. For convenient, let

$$X_o = X \setminus \cup_{i=1}^{n} \overline{P}_i. \tag{118}$$

When $X = \mathbf{R}^n$ with the Euclidean metric, the above problem can be equivalently formulated as an optimal control problem in which

$$\dot{x}(t) = u(t), \quad t \in [t_0, t_f], \quad x(t) \in \mathbf{R}^n, \quad u(t) \in \mathbf{R}^n, \tag{119}$$

with path constraints

$$x(t_0) = x, \quad x(t_f) = y, \quad \phi_i(x(t)) \geq 0, \quad t \in [t_0, t_f], \quad 1 \leq i \leq N. \tag{120}$$

Here $\phi_i(x)$ is the level set function of each obstacle.
The cost is the length of the path

$$J(u) = \int_{t_0}^{t_f} |u(t)| \, \mathrm{d}t. \tag{121}$$

As a result, if under certain condition, the shortest path is separable, we can apply method of evolving junctions to solve the problem. The separability is addressed in the next section.

4.3 Separability of the Shortest Path in $\mathbf{R}^n$

In this section, we prove the following theorem concerning the separability of the shortest path. Let R_i denote the boundary of P_k.

Theorem 15 *Let the boundaries R_i of the obstacles be piecewise C^2 and the total number of points of C^2-discontinuity is finite. Let γ_{opt} be an optimal solution to the shortest path problem. Then there exist intervals $\{I_k \subset [0,1]\}$ such that every $\gamma_{opt}(t)|_{t \in I_k}$ is on the boundary of one obstacle. Outside these intervals, $\gamma(t)$ is a union of straight line segments. Moreover, each line segment is tangent to the obstacles.*

We validate this claim by defining a new metric in the region such that the metric inside the obstacles can be arbitrarily large. We show that the shortest path in the new metric can be arbitrarily close to the shortest path of the original problem. On the other hand, the shortest path in the new metric is a geodesic whose structure is described in the theorem.

For simplicity, we assume the boundaries of all obstacles to be C^2 in this section.

4.3.0.1 A new metric

Define a continuous function $g\colon \mathbf{R} \to \mathbf{R}$ as follows:

$$g(x) = \begin{cases} B, & x < -\epsilon, \\ \text{smooth and decreasing}, & -\epsilon \leq x < 0 \\ 1, & x >= 0. \end{cases} \tag{122}$$

Here B is a large number which will be determined later.

Now define the following Riemannian metric: at each $p \in M$, $g_p\colon T_pM \times T_pM \to \mathbf{R}$ is

$$g_p(\mathbf{x}, \mathbf{y}) = g(d(p, \Gamma))^2 \langle \mathbf{x}, \mathbf{y} \rangle .$$

Here, Γ is the union of the boundaries of all obstacles, $d(p, \Gamma)$ is the signed distance between p and Γ, and $\langle \mathbf{x}, \mathbf{y} \rangle$ is the usual inner product in $\mathbf{R}^2$.

We note that M is a smooth manifold endowed with this metric. For any feasible curve γ, we denote $\mathbb{L}_{\text{new}}(\gamma)$ and $\mathbb{L}_{\text{old}}(\gamma)$ the length of γ under the the new metric and the old metric(the Euclidean metric) respectively. For example, if $\gamma\colon [0, 1] \to M$ is C^2,

$$\mathbb{L}_{\text{new}}(\gamma) = \int_0^1 |\gamma'(t)| g(d(\gamma(t), \Gamma))\, \mathrm{d}t.$$

4.3.0.2 The structure of the optimal path

Let G denote the set of paths connecting X and Y in M. Let $\alpha\colon [0,1] \to M$ be the shortest path in G under the new metric, i.e.

$$\alpha(t) = \text{argmin}_{\gamma \in G}\, \mathbb{L}_{\text{new}}(\gamma).$$

Since M is a smooth Riemannian manifold, α is a geodesic in M [23]. Therefore, α is straight line segment outside the obstacles.

For each point x on the boundary of P_i, we can associate it with a point inside P_i in the normal line with a distance of ϵ to x. All those points form another C^2 curve. Denote the domain enclosed by such curves by P_i^{ϵ}. Similarly, we define $P_i^{2\epsilon}$ and hence $P_i^{2\epsilon} \subset P_i^{\epsilon} \subset P_i$. It is not hard to show that ∂P_i^{ϵ} is simple provided ϵ is sufficiently small. Also we have the following:

Lemma 16 *There exists a constant B in* (122) *such that α is entirely outside $P_i^{2\epsilon}$.*

Proof 2 *For any $p, q \in \partial P_i^{\epsilon}$, denote $\overline{pq}$ the line segment connecting p, q. Let*

$$S = \{\, (p,q) \in \partial P_i^{\epsilon} \times \partial P_i^{\epsilon} \mid \overline{pq} \cap P_i^{2\epsilon} \neq \emptyset \,\}.$$

S is compact since S is equivalent to

$$S = \{\, (p,q) \in \partial P_i^{\epsilon} \times \partial P_i^{\epsilon} \mid d(\overline{pq}, P_i^{2\epsilon}) \leq 0 \,\}$$

and d is continuous. Moreover, the function $h(p,q) = \mathbb{L}_{\text{old}}(\overline{pq})$ where $(p,q) \in S$ is always positive. Therefore, h has a positive minimum l. Choose any path in the set of feasible paths F, denote its length under the new metric by L. Select B such that $lB > L$. Now for any two points p, q on ∂P_i^{ϵ}, if the line segment connecting them intersects with $P_i^{2\epsilon}$, then the length under the new metric should be greater than $lB > L$. This implies that α is outside $P_i^{2\epsilon}$ everywhere since α is straight in P_i^{ϵ}.

Intuitively, when B becomes larger, the penalty imposed on the portion inside obstacles also becomes larger. Hence in order to reduce the length, the path mustn't pass through obstacles too much. Now restrict α on the interval $[S,T]$ where $\alpha(S), \alpha(T) \in \partial P_i$ and the image of α $\operatorname{Img}\alpha \subset P_i$ for $t \in [S,T]$. Denote $R = \alpha(S), S = \alpha(T)$. See the figure below.

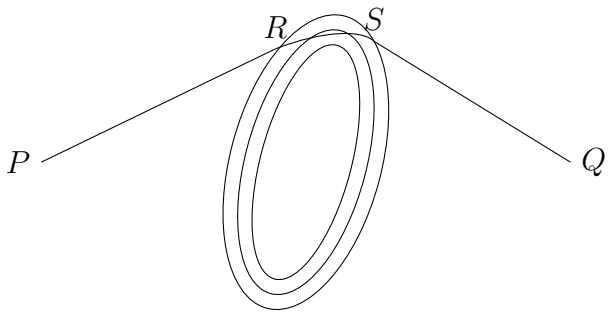


Figure 3: An illustration of α.

Denote $\tilde{\beta}(t)$ the arc length parametrization of ∂P_i between R and S and $\tilde{\alpha}(t)$ the part of α between R and S.

Lemma 17

$$\mathbb{L}_{\text{old}}(\tilde{\alpha}(t)) \geq (1 - 4\epsilon\kappa)\, \mathbb{L}_{\text{old}}(\tilde{\beta}(t)) \tag{123}$$

where κ is the maximal curvature.

Proof 3 *First we show that $\tilde{\alpha}(t)$ can be expressed as*

$$\tilde{\alpha}(t) = \tilde{\beta}(t) + \epsilon(t)\mathbf{n}(t),$$

where $\epsilon(t) < 2\epsilon$ and $\mathbf{n}(t)$ is the unit inward normal, i.e, the geodesic α will intersect with the normal line only once. If not, for any $t \in [0,1]$, let

$$I(t) = \{\, s \in [0,1] \mid (\tilde{\alpha}(s) - \tilde{\beta}(t)) \cdot \tilde{\beta}'(t) = 0 \,\}$$

and $t_0 = \inf I(t), t_1 = \sup I(t)$. Since $\tilde{\alpha}(1) = S$ is not on the normal line, we have $t_1 < 1$. Moreover, it is easy to see that $t_0, t_1 \in I(t)$ by the continuity of $(\tilde{\alpha}(s) - \tilde{\beta}(t)) \cdot \tilde{\beta}'(t)$.

Thus $\tilde{\alpha}(s), s \in [t_0, t_1]$ must be entirely on the normal line due to its optimality. Now choose Δt small enough such that $\overline{\tilde{\alpha}(t_0)\tilde{\alpha}(t_1+\Delta t)}$ is in the region formed by $\tilde{\alpha}$ and $\tilde{\beta}$. But $\overline{\tilde{\alpha}(t_0)\tilde{\alpha}(t_1+\Delta t)}$ has smaller length, which is a contradiction.

This representation of $\tilde{\alpha}$ gives

$$\frac{d\tilde{\alpha}(t)}{dt} = \frac{\tilde{\beta}(t)}{dt} + \epsilon(t)\frac{d\mathbf{n}(t)}{dt} + \epsilon'(t)\mathbf{n}(t)$$

and

$$\begin{aligned}\left|\frac{d\tilde{\alpha}}{dt}\right|^2 &= (\tilde{\beta}'(t), \tilde{\beta}'(t)) + \epsilon(t)^2 \left|\mathbf{n}'(t)\right|^2 + \epsilon'(t)^2 + 2(\tilde{\beta}'(t), \epsilon(t)\mathbf{n}'(t)) \\ &\quad + 2(\tilde{\beta}'(t), \epsilon'(t)\mathbf{n}(t)) + 2\epsilon(t)\epsilon'(t)(\mathbf{n}'(t), \mathbf{n}(t)) \\ &= 1 + \epsilon(t)^2 \left|\mathbf{n}'(t)\right|^2 + \epsilon'(t)^2 + 2\epsilon(t)(\tilde{\beta}'(t), \mathbf{n}'(t)) \\ &\geq 1 - 2\epsilon(t)\kappa \geq 1 - 4\epsilon\kappa \geq (1-4\epsilon\kappa)^2\end{aligned}$$

for sufficiently small ϵ. Consequently,

$$\mathbb{L}_{\text{old}}(\tilde{\alpha}(t)) \geq \mathbb{L}_{\text{old}}(\tilde{\beta}(t)) - 4\epsilon\kappa\, \mathbb{L}_{\text{old}}(\tilde{\beta}(t)).$$

The constructed path $\alpha(t)$ may intersect with P_i multiple or even infinite times. However, its total length can be controlled. More specifically, we have the following

Lemma 18 *Suppose $\tilde{\alpha}_1(t), t \in [0,1]$ and $\tilde{\alpha}_2(t), t \in [0,1]$ are arc length parametrizations of two segments of α inside the same obstacle, and $\tilde{\beta}_1$, $\tilde{\beta}_2$ are two curves defined as in* (123)*, i.e.*

$$\tilde{\alpha}_1(t) = \tilde{\beta}_1(t) + \epsilon_1(t)\mathbf{n}_1(t),$$

$$\tilde{\alpha}_2(t) = \tilde{\beta}_2(t) + \epsilon_2(t)\mathbf{n}_2(t),$$

Then either $\text{Img}(\tilde{\beta}_1) \subset \text{Img}(\tilde{\beta}_2)$ or $\text{Img}(\tilde{\beta}_1) \cap \text{Img}(\tilde{\beta}_2) = \emptyset$. Moreover, if the former is true, then

$$\mathbb{L}_{\text{old}}(\tilde{\beta}_1) < \frac{\epsilon_1}{2-\epsilon_1}\, \mathbb{L}_{\text{old}}(\tilde{\beta}_2).$$

where $\epsilon_1 = 4\epsilon\kappa$. In other words, the length decreases exponentially.

Proof 4 *The first part of the claim is obvious. For the second part, notice*

$$\mathbb{L}_{\text{old}}(\tilde{\alpha}_1(t)) \geq \mathbb{L}_{\text{old}}(\tilde{\beta}_1(t)) - 4\epsilon\kappa\,\mathbb{L}_{\text{old}}(\tilde{\beta}_1(t))$$

$$\mathbb{L}_{\text{old}}(\tilde{\alpha}_2(t)) \geq \mathbb{L}_{\text{old}}(\tilde{\beta}_2(t)) - 4\epsilon\kappa\,\mathbb{L}_{\text{old}}(\tilde{\beta}_2(t)).$$

By the definition of α, we have

$$\mathbb{L}_{\text{old}}(\tilde{\beta}_2) - \mathbb{L}_{\text{old}}(\tilde{\beta}_1) > (\mathbb{L}_{\text{old}}(\tilde{\beta}_2) + \mathbb{L}_{\text{old}}(\tilde{\beta}_1))(1-\epsilon_1)$$

Therefore,

$$\mathbb{L}_{\text{old}}(\tilde{\beta}_1) < \frac{\epsilon_1}{2-\epsilon_1}\,\mathbb{L}_{\text{old}}(\tilde{\beta}_2).$$

Proof of the structure theorem

Let β be the curve consisting of the straight part of α and all the $\tilde{\beta}(t)$ parts as above, i.e. portions of the boundaries of P_i and γ_{opt} the shortest path of our original problem, i.e.

$$\gamma_{opt}(t) = \text{argmin}_{\gamma \in F}\,\mathbb{L}_{\text{old}}(\gamma).$$

First we show that $\mathbb{L}_{\text{old}}(\beta)$ and $\mathbb{L}_{\text{old}}(\gamma_{opt})$ can be arbitrarily close. In fact, by summing the inequality in (123), we get

$$\mathbb{L}_{\text{old}}(\alpha) \geq \mathbb{L}_{\text{old}}(\beta) - 4C\epsilon\kappa L$$

where L is the total perimeters of all the obstacles and $C = \sum_0^\infty (\frac{\epsilon_1}{2-\epsilon_1})^i$. This implies

$$\mathbb{L}_{\text{old}}(\beta) \geq \mathbb{L}_{\text{old}}(\gamma_{opt}) = \mathbb{L}_{\text{new}}(\gamma_{opt}) \geq \mathbb{L}_{\text{new}}(\alpha) \geq \mathbb{L}_{\text{old}}(\alpha) \geq \mathbb{L}_{\text{old}}(\beta) - 4C\epsilon\kappa L.$$

In other words,

$$|\mathbb{L}_{\text{old}}(\gamma_{opt}) - \mathbb{L}_{\text{old}}(\beta)| \leq 2C\epsilon\kappa L.$$

As a result, we only need to minimize $\mathbb{L}_{\text{old}}(\beta)$. By the construction of β, it can be represented by a series of points $(x_0, x_1, \ldots, x_{n+1})$ and

$$2\mathcal{L}(\beta) = \sum_{i=1}^{n} J(x_i).$$

For any i, x_i minimizes $J(x_i)$ only if

$$\nabla J(x_i) = (\frac{x_i - x_i^s}{\|x_i - x_i^s\|} \cdot \mathbf{T})\mathbf{T} + \text{sign}(d^+(x_i, x_i^c) - d^-(x_i, x_i^c))\mathbf{T} = 0.$$

But

$$\left\| \frac{x_i - x_i^s}{\|x_i - x_i^s\|} \cdot \mathbf{T} \right\| \leq 1$$

which implies that $x_i - x_i^s$ is parallel to $\mathbf{T}$, i.e. $x_i - x_i^s$ is tangent to P_k. Therefore, if β minimizes $\mathcal{L}$, then all the straight parts of β should be tangent to a obstacle. By the inequality above, we know that this is also the solution to the original problem.

4.4 Shortest Path in R^2

4.4.1 Method of Evolving Junctions

From theorem 15, the shortest path is separable. Therefore, we can apply the method of evolving junctions to find the shortest path. Let $(x_0, x_1, \cdots, x_n, x_{n+1})$ be the set of junctions. Notice we have remove the time dependence of each junction. Also let n_i be the index of the obstacle on which x_i sits. The shortest distance in the free space is simply the Euclidean distance between them, i.e.

$$J_0(x_k, x_{k+1}) = \|x_k - x_{k+1}\|. \tag{124}$$

To find the shortest path between two junctions, notice as shown in Figure 4, for each point x_i there are two cases on how it is connected to the junctions before and after. In the first case (the left picture), x_{i-1} and x_i are connected by a straight line, and x_{i+1} is connected to x_i by a curve on the boundary. In this case, we denote x_{i-1} by x_i^s and x_{i+1} by x_i^c. The sup-index s (or c) represents that the two points are connected by a straight line (or curve). In the second case, as shown by the right picture of Figure 4, we have $x_i^c = x_{i-1}$ and $x_i^s = x_{i+1}$. These notations can be extended to all connecting points including x_0 and x_{n+1}, if we assume $x_0^c = x_0, x_{n+1}^c = x_{n+1}$. In both cases, x_i and x_i^c divide the boundary R_{n_i} into two parts, R_i^+ and R_i^-, where R_i^+ is the arc from x_i to x_i^c with the counterclockwise direction and R_i^- the clockwise direction.

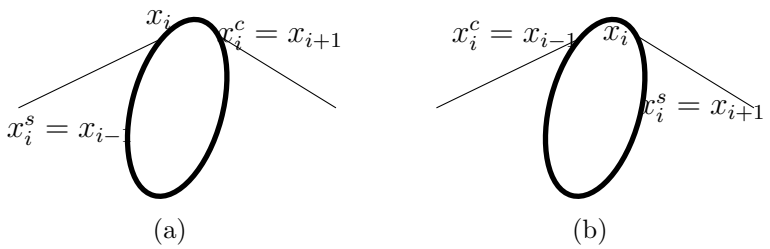


Figure 4: Each junction is connected to the junctions before and after it by a straight line segment or an arc of the boundary.

Because R_i is rectifiable, the arc lengths of R_i^+ and R_i^{-1}, denoted by $d_i^+(x_i, x_i^c)$ and $d_i^-(x_i, x_i^c)$ respectively, are finite. The distance between x_i and x_i^c along the boundary is then

$$J_{n_i}(x_i, x_i^c) = \min\left\{d_i^+(x_i, x_i^c), d_i^-(x_i, x_i^c)\right\}.$$

Furthermore, for each x_i, if we define

$$J(x_i) = J_0(x_i, x_i^s) + J_{n_i}(x_i, x_i^c). \tag{125}$$

Then the length of path $\gamma \in H$ is

$$J(\gamma) = J(x_1, x_2, \cdots, x_n) = \frac{1}{2}\sum_{i=0}^{n+1} J(x_i) \tag{126}$$

4.4.1.1 Gradient descent

Proposition 19 *The gradient flow corresponding to the length functional J is*

$$\frac{dx_i}{dt} = -\nabla J(x_1, \cdots, x_i, \cdots, x_n) = -\nabla J(x_i). \tag{127}$$

Moreover, if R_i is C^2, then

$$\nabla J(x_i) = (\frac{x_i - x_i^s}{\|x_i - x_i^s\|} \cdot \mathbf{T})\mathbf{T} + \text{sign}(d^+(x_i, x_i^c) - d^-(x_i, x_i^c))\mathbf{T} \tag{128}$$

where $\mathbf{T}$ is the unit tangent at x_i to the boundary R_i in counter-clockwise direction.

Proof 5 *Let $p \in R_i$ and $T_p(R_i)$ be the tangent space of R_i at p. By definition, ∇J satisfies*

$$\langle \nabla J, \mathbf{X} \rangle_{x_i} = \mathbf{X}(J)_{x_i}, \quad \forall x_i \in R_{n_i}, \mathbf{X} \in T_{x_i}(\mathbf{R}_{n_i}) \tag{129}$$

Let $\mathbf{X} = \mathbf{T}$ *and* $r(t), t \in [0,1]$ *be the arc length parametrization of* R_{n_i} *with* $r(0) = x_i$ *and* $r'(0) = \mathbf{T}$*. By the definition of* J *and a direct computation, we have the following for* $t = 0$*:*

$$\frac{dJ(r(0))}{dt} = \frac{x_i - x_i^s}{\|x_i - x_i^s\|} \cdot \mathbf{T} + \text{sign}(d^+ - d^-). \tag{130}$$

Combining (129) *and* (130)*, we get equation (*(127)*).*

Remark 3 *The computation of* $\nabla J(x_i)$ *also holds if we only assume* R_i *to be piecewise* C^2*. At a point of* C^2 *discontinuity, we can choose* $\mathbf{T}$ *to be either the left tangent or the right tangent vector.*

4.4.1.2 Intermittent Diffusion

The gradient flow (127) provides an explicit formula to move the points x_i on the boundaries. Their steady states includes all local optimal positions which define the locally optimal path. The number of steady states could be large in the shortest path problem. In certain situations, this can be estimated. For example, let the obstacles be smooth. If we have $N \geq 1$ obstacles and $2N$ junctions, then the length functional J is actually a smooth scalar-valued function on a torus of dimension $2N$, i.e.

$$J \colon T^{2N} \to \mathbf{R}.$$

If J is a Morse function (i.e., all critical points of $\mathcal{L}$ are non-degenerate), then there are at least 2^{2N} distinct critical points [15]. Each minimal point defines a minimum path. Obviously, the exponential growth of the number of critical points imposes a great challenge on the gradient descent method. Furthermore, as the number of connecting points changes, the dimension of the torus (phase space of the gradient flow (127)) also changes. Thus, a global optimization technique must be used to find a globally optimal path. In this paper, we adopt the intermittent diffusion strategy to address this problem.

For our problem, since the points are moving on the boundaries, we add one dimensional noise to the gradient flow as follows:

$$dx_i = -\nabla J(x_i)dt + \sigma(t)\mathbf{T}dW(t). \tag{131}$$

4.4.1.3 Algorithm

Now, we are ready to present our algorithm:

Shortest Path Problem in $\mathbf{R}^n$

Input: level set function of obstacles $\phi(x)$,

starting and ending points x and y,

number of intermittent diffusion intervals m.

Output: The shortest path γ_{opt} represented by a set of junctions.

---

1. Initialization. Find the initial path $\gamma^{(0)} = (x_0, \cdots, x_{n+1})$ which consists of all the intersection points $\overline{xy}$ with the boundary of the obstacles;
2. Select duration of diffusion $\Delta T_l, l \leq m$;
3. Select diffusion coefficients $\sigma_l, l \leq m$;
4. **for** $l = 1 : m$
5. $\quad \gamma^{(l)} = \gamma^{(0)}$;
6. $\quad$ **for** $j = 1 : \Delta T_l$
7. $\quad\quad$ Update $\gamma^{(l)}$ according to (131) with $\sigma(t) = \sigma_l$;
8. $\quad\quad$ Remove junctions from or add junctions to $\gamma^{(l)}$ when necessary;
9. $\quad$ **end**
10. $\quad$ **while** $\|\nabla^c J(\gamma^{(l)})\| > \epsilon$
11. $\quad\quad$ Update $\gamma^{(l)}$ according to (127);
12. $\quad\quad$ Remove junctions from or add junctions to $\gamma^{(l)}$ when necessary;
13. $\quad$ **end**

14. **end**

15. Compare $J(\gamma^{(l)}), l \leq m$ and set $\gamma_{opt} = \text{argmin}_{l \leq m} J(\gamma^{(l)})$;

In the algorithm, the moving points x_i are updated only on the boundaries of obstacles, which are represented by their level set functions in our implementation. The boundary of an obstacle is the zero level set of a signed distance function. We achieve this by projecting each step of solving (131) and (127) on the zero level set of the obstacles. In addition, we must compute the intersections of the line segments with the boundaries of obstacles. We accomplished this in the same level set framework. To solve (131) and (127), different schemes can be used. In the next section, we give our numerical implementations of each step in the algorithm in detail.

4.4.2 Numerical Implementation

We use a level set representation, the signed distance function [50], to implicitly express the boundaries of the obstacles. More precisely, let $p_i(x)$ be the distance from point x to the boundary of obstacle P_i, i.e. $p_i(x) = \min_{y \in \partial P_i} \|x - y\|$. The signed distance function $\phi_i(x)$is then defined as

$$\phi(x) = \begin{cases} p(x), & x \text{ is outside } P; \\ -p(x), & x \text{ is inside } P. \end{cases} \tag{132}$$

Under this representation, the outward unit normal direction at x on the boundary ∂P is simply

$$\mathbf{n} = \nabla \phi \tag{133}$$

and the curvature at x can be computed by

$$\kappa = \nabla \cdot \nabla \phi. \tag{134}$$

The connecting points should move only on the boundaries. To ensure this, we use the following lemma to project the updates along the tangent directions to the boundaries as shown in Figure 5.

Lemma 20 *Let α be a planar curve, $\mathbf{T}$ and $\mathbf{n}$ are the tangent and normal directions at one point x on α. l is a line that is parallel to $\mathbf{n}$, and l intersects with α and $\mathbf{T}$ at P and Q respectively. Let h denote the lengths of $\overline{PQ}$, d the length of $\overline{xQ}$, and κ the curvature at x, then*

$$|\kappa| = \lim_{d\to 0} \frac{2h}{d^2}.$$

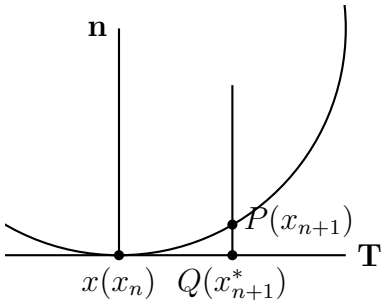


Figure 5: The projection from the tangent direction to the boundary used in Lemma 20 and Equation (136).

Proof 6 *Let's assume α is arc-length parametrized and $x = \alpha(0)$. In the neighborhood of x, α has Taylor expansion*

$$\alpha(t) = x + t\mathbf{T} + \frac{t^2}{2}\kappa\mathbf{n} + \mathbf{o}(t^2).$$

Therefore, $P = x + d\mathbf{T} + d^2/2\kappa\mathbf{n} + \mathbf{o}(d^2)$, $Q = x + d\mathbf{T}$, hence

$$\lim_{d\to 0} \frac{2h}{d^2} = \lim_{d\to 0} \frac{d^2|\kappa| + \mathbf{o}(d^2)}{d^2} = |\kappa|.$$

This lemma enables us to project the gradient flow from the tangent space to the manifold very easily. For the convenience of the presentation, let us denote

$$f(x) = -(\frac{x_i - x_i^s}{\|x_i - x_i^s\|} \cdot \mathbf{T}) - \text{sign}(d^+(x_i, x_i^c) - d^-(x_i, x_i^c))$$

Then (127) becomes

$$\frac{dx}{dt} = f(x)\mathbf{T}. \tag{135}$$

This can be computed by evolving the points in the tangent space followed by a projection to the boundary. More precisely, as shown in Figure 5, we first compute x^*_{n+1} in the tangent space by

$$x^*_{n+1} - x_n = f(x_n)\Delta t\mathbf{T}.$$

Then the projected point x_{n+1} is the point on the boundary such that $x_{n+1} - x^*_{n+1}$ is parallel to $\mathbf{n}$. By lemma (20),

$$\|x_{n+1} - x^*_{n+1}\| = \frac{1}{2}|\kappa|\|x_n - x^*_{n+1}\|^2 = \frac{1}{2}|\kappa|(f(x_n)\Delta t)^2. \tag{136}$$

The direction of $x_{n+1} - x^*_{n+1}$ depends on the sign of the curvature. It is easy to see the direction is $-\operatorname{sign}(\phi(x_n))\mathbf{n}$. Hence, we have

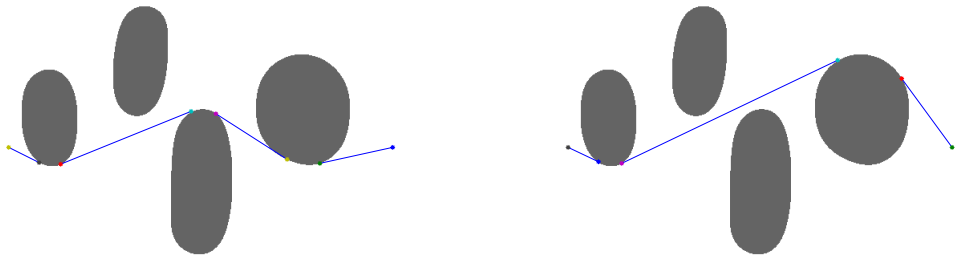
$$x_{n+1} - x^*_{n+1} = -\operatorname{sign}(\phi(x_n))\frac{|k|(f(x_n)\Delta t)^2}{2}\mathbf{n}. \tag{137}$$

We remark that the projection from the tangent space to the boundary can be accomplished in other ways, which may not depend on the curvature. This is appropriate especially when the boundaries are not smooth. The performance of the algorithm is similar when using different projection methods. One particular case is if arc length parametrization is used in the computation, one can add noise directly to the parameter without using projections.

To discretize (131), let x^*_{n+1} be the point along the tangent direction such that

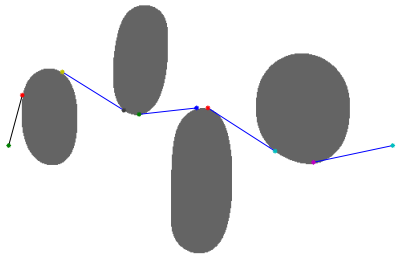
$$x^*_{n+1} - x_n = f(x_n)\Delta t\mathbf{T} + \sigma_n\sqrt{\Delta t}\xi\mathbf{T}, \tag{138}$$

where $\xi \in N(0,1)$ is a standard normal random variable. σ_n here is also chosen to be random within a range $[a,b]$. The range depends on the size of the obstacles. If the obstacle is large, large σ needs to be selected in order for the path to jump out of the local traps. For the examples below, σ is taken in $[1,2]$.

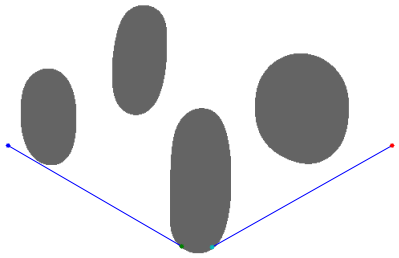


(a) $L = 1.0264$, 26 times

(b) $L = 1.0966$, 6 times



(c) $L = 1.1320$, 9 times



(d) $L = 1.1501$, 9 times

Figure 6: Example 1, the algorithm finds 4 different shortest paths in one trials.

The arc lengths $d^+(x_i, x_i^c)$ and $d^-(x_i, x_i^c)$ can be computed in the same manner, i.e., solve (135) with $f(x) = 1$ or $f(x) = -1$ respectively with initial condition $x(0) = x_i$, record the time t^+ or t^- it hits x_i^c, then we have $d^+(x_i, x_i^c) = t^+\Delta t$ and $d^-(x_i, x_i^c) = t^-\Delta t$.

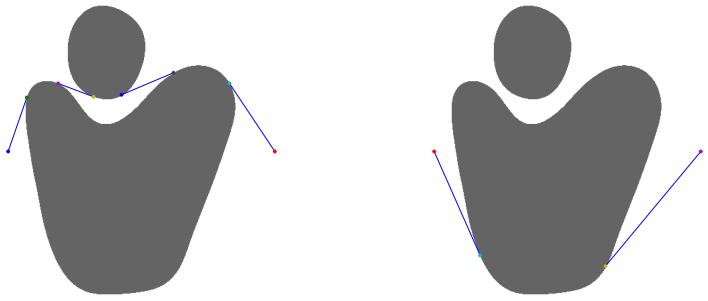
4.4.3 Numerical Results

In this section, we illustrate the performance of our method by showing the following examples.

Example 1. There are four obstacles in this example shown in Figure 6. The starting point and the ending point are $X = [0.5, 0.02]$ and $Y = [0.5, 0.98]$ respectively. We choose $N = 50$ and obtain 50 sample paths. The algorithm finds 4 different locally minimal paths as shown in Figure 6. Among them (14a) is the global minimal path. And it is observed 26 times, which is far more than the other locally optimal solutions.

Example 2. In this example, there are two obstacles which form a tunnel in Figure 7. This is a non-convex case. We choose $X = [0.5, 0.02], Y = [0.5, 0.98]$. The algorithm finds two local minimal paths with the global minimizer passing through

the tunnel as shown in Figure (7a). The algorithm finds the global minimizer 46 times, which is much more frequent than visiting the other local minima.

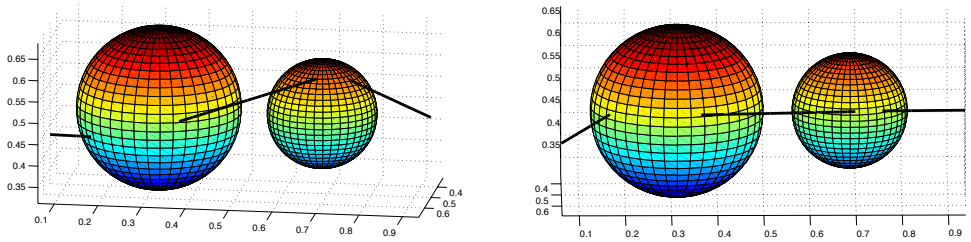


(a) $L = 1.1874$, 46 times (b) $L = 1.3393$, 4 times

Figure 7: Example 2, the algorithm finds 2 different shortest paths in one trial.

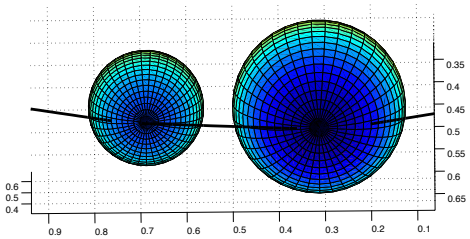
Although our algorithm is presented for 2-D cases, the idea can be extended to 3 or more dimensions with minor modifications. The main change is that the general implementation in higher dimensions needs the shortest length between two moving points on the surface which can be computed by the fast marching method [63] on the surface defined by the boundary of an obstacle. Here, we show an example in 3-D.

Example 3. In this example, the obstacles consist of two balls: one is centered at $[0.5, 5/16, 0.5]$ with radius $3/16$ and the other one is centered at $[0.5, 11/16, 0.5]$ with radius $1/8$. The starting point and the ending point are $X = [0.5, 1/16, 0.5 - \sqrt{3}/24]$ and $Y = [0.5, 15/16, 0.5]$ respectively. The algorithm finds three local minimizers in 30 runs as shown in Figure 8 and (8c) is the global minimal path. It was visited 27 times, which dominants the frequency of appearance.



(a) Occurs once

(b) Occurs twice



(c) Occurs 27 times

Figure 8: Example 3, the algorithm finds 3 different shortest paths in one trial.

4.5 Shortest Path with Polyhedra Obstacles

4.5.1 Method of Evolving Junctions

In this section, we focus on applying the method of evolving junctions to the shortest path problem with polyhedral obstacles in $\mathbf{R}^3$. The restriction on polyhedral obstacles allows us to achieve further dimension reductions. For obstacles with smooth boundaries, the implementation requires computations of geodesics on the boundaries between two given points. In [17], this is achieved by either traversing the boundaries in $\mathbf{R}^2$, or fast marching on the boundary surfaces in $\mathbf{R}^3$. However, for polyhedral obstacles, the geodesics also has a similar simple structure, i.e. the geodesic between two points on a polyhedron is a conjunction of line segments whose ending points

are located on polyhedron edges. And to determine the geodesic is equivalent to determine those junction points. Therefore the overall shortest path connecting X and Y is merely a conjunction of line segments whose ending points lie on obstacle edges. In other words, the domain for each junction is an 1-D interval. This makes the algorithm extremely simple and efficient.

A feature of this study is that we do not restrict the obstacles to be convex polyhedrons. The algorithm we develop can equally be applied to non-convex polyhedrons. For polyhedrons with Euler characteristic 2, which include all convex polyhedrons and concave polyhedrons without holes, our algorithm can find the globally optimal path with probability arbitrarily close to 1 in a finite time. However, when dealing with more sophisticated polyhedrons, for example, polyhedrons with complicated holes, certain topological problems emerge, and prevent us from obtaining the globally optimal path. We will discuss this issue at the end of the paper as well as some possible solutions.

Let $\{P_k\}_{k=1}^N$ be N polyhedral obstacles in $\mathbf{R}^3$. Each obstacle P_k is determined uniquely by its vertices, edges and faces. Denote V, E, F the set of vertices, edges and faces of P_k respectively. We do not limit the polyhedrons to be convex. However, we will focus on polyhedrons without holes in this section, i.e. polyhedrons whose Euler characteristic is 2. The Euler characteristic is defined by

$$\chi = |V| - |E| + |F|.$$

Polyhedrons with holes will be discussed in the last section. For any edge $e \in E$, it has a representation

$$e = (\mathbf{u}, \mathbf{v})$$

where $\mathbf{u}, \mathbf{v}$ are the coordinates of the ending points of e. Any point x on edge $e = (\mathbf{u}, \mathbf{v})$ can then be represented by the following expression

$$x(\mathbf{u}, \mathbf{v}, \theta) = \theta\mathbf{u} + (1 - \theta)\mathbf{v}. \tag{139}$$

Thus to determine the position of a point on an edge, one only needs to find its corresponding θ.

As in the last section, the cost in the free space is

$$J_0(x, y) = \|x - y\|. \tag{140}$$

For any two points x, y on the edges of P_k, we can define the distance $J_k(x, y)$ between them to be the length of the shortest path on P_k connecting x and y. If we view P_k as a surface in $\mathbf{R}^3$, i.e. a two dimensional manifold, then $J_k(x, y)$ is nothing but the geodesic on P_k connecting x and y. For instance, for any x and y in a tetrahedron, $J_k(x, y) = \|x - y\|$ since the line segment joining them is on the surface. For general polyhedrons, the shortest path is composed by a sequence of line segments connected to each other. To be more specific, the shortest path can be represented by $(x_0, x_1, x_2, \ldots, x_{n_k}, x_{n_k+1})$ where $x_0 = x, x_{n_k+1} = y$ and each x_i is a point on some edge $e_i = (\mathbf{u}_i, \mathbf{v}_i)$. The shortest distance $J_k(x, y)$ therefore equals

$$J_k(x, y) = J(x_1, x_2, \ldots, x_{n_k}) = \sum_{i=0}^{n_k} J_0(x_i - x_{i+1}).$$

Denote $x_i = \theta_i \mathbf{u}_i + (1 - \theta_i)\mathbf{v}_i$, we then have

$$\begin{aligned} J(\theta_1, \ldots, \theta_{n_k}) &= J(x_1, \ldots, x_{n_k}) \\ &= \sum_{i=0}^{n_k} \|\theta_{i+1}\mathbf{u}_{i+1} + (1 - \theta_{i+1})\mathbf{v}_{i+1} - \theta_i \mathbf{u}_i - (1 - \theta_i)\mathbf{v}_i\|. \end{aligned}$$

Because all the obstacles here are polyhedrons, the paths on the boundaries of the obstacles also consist of a sequence of line segments connected by points on the edges. Therefore, by putting all the connecting points together and relabeling them, the shortest path connecting X and Y can be represented by $(x_0, x_1, x_2, \ldots, x_n, x_{n+1})$ where $x_0 = X, x_{n+1} = Y$.

Let us denote

$$J(x_i) = \|x_{i-1} - x_i\| + \|x_{i+1} - x_i\|. \tag{141}$$

Then the length of the path is

$$J(x_1, \ldots, x_n) = \frac{1}{2} \sum_{i=1}^{n} J(x_i). \tag{142}$$

Again all x_is are on the edges of the obstacle. Denote $x_i = \theta_i \mathbf{u}_i + (1 - \theta_i)\mathbf{v}_i$, $J(x_i)$ then becomes

$$J(\theta_i) = \|\theta_i \mathbf{u}_i + (1 - \theta_i)\mathbf{v}_i - x_{i-1}\| + \|\theta_i \mathbf{u}_i + (1 - \theta_i)\mathbf{v}_i - x_{i+1}\|. \tag{143}$$

To find the optimal path, we differentiate $J(\theta_i)$ with respect to θ_i to obtain

$$\nabla J(\theta_i) = \frac{(x_i - x_{i-1}) \cdot (\mathbf{u}_i - \mathbf{v}_i)}{\|x_i - x_{i-1}\|} + \frac{(x_i - x_{i+1}) \cdot (\mathbf{u}_i - \mathbf{v}_i)}{\|x_i - x_{i+1}\|}. \tag{144}$$

and the method of evolving junctions solves

$$\frac{d\theta_i}{dt} = -\nabla J(\theta_i) + \sigma(t) dW(t). \tag{145}$$

we have the following theorem

Theorem 21 *If all the obstacles have Euler characteristic 2, then for any small number ϵ,equation* (145) *converges to the global minimizer with probability $1 - \epsilon$ for suitable $\sigma(t)$.*

We will postpone the proof until the last section when we discuss the topological issues.

4.5.2 Numerical Implementation

In this section, we discuss how to solve equation (145).

Discretization of SDE We use forward Euler method to discretize equation (145) as following

$$\frac{\theta_i^{j+1} - \theta_i^j}{\Delta t} = -\nabla J(\theta_i^j) + \sigma(j\Delta t)\sqrt{\Delta t}\xi$$

where $\xi \sim N(0, 1)$ is a normal random variable.

Initialization We can use the optimal path whose junctions are restricted on vertices of the obstacles to initialize the path. This initialization can be obtained efficiently by a method called visibility graph . The visibility graph W is a weighted graph whose nodes are the vertices of all the obstacles as well as the starting and ending points X, Y, and there is an edge between vertices $\mathbf{u} \in W$ and $\mathbf{v} \in W$ if and only if they are visible to each other, that is, if the line segment $\overline{\mathbf{uv}}$ doesn't intersect with any obstacles. The weight of edge $\mathbf{uv}$ is simply the Euclidean distance of $\overline{\mathbf{uv}}$. One thing to notice is that the visibility graph we construct here is essentially 2D, in the sense that it encodes whether two points are visible to each other. This is fundamentally different from the 3D reduced visibility graph (3DRVG) in the introduction. 3DRVG consists of connected planes as opposed to straight line segments which becomes complicated when there are more than one obstacles. See [32]. After the visibility graph is constructed, the initialization is the shortest path between X and Y on the visibility graph W which can be obtained efficiently by Dijkstra algorithm..

Evolution when a junction reaches a vertex In the proposed method, the junctions move according to the SDEs if they are on the interior of edges. When a junction $x = (\mathbf{u}, \mathbf{v}, \theta)$ reaches to a vertex $\mathbf{u}$ following the gradient flow, it continues moving according to different rules depending on whether the two neighbors of x are on the same obstacle or not. If the neighbors of x are both on the same obstacle as x, we call x an interior junction, otherwise we call x an exterior junction. In other words, an exterior junction is one of the two ending points of the line segments that connects two different obstacles. The following are the rules for interior and exterior junctions reaching the vertices respectively.

Case 1. $x = (\mathbf{u}, \mathbf{v}, \theta = 1)$ is an interior junction. See the following illustration where (x_1, x_2, x, x_4) is the path on the obstacle and x_1, x_4 are exterior junctions. When x hits $\mathbf{u}$ ($\theta = 1$), path $(x_1, u = x, x_4)$ will have smaller length than $(x_1, x_2, u =$

x, x_4) length. In other words, all the junctions adjacent to **u** will be dragged to **u** except the exterior junctions. Hence we remove all the junctions adjacent to **u** and add junctions on the edges adjacent to **u** that haven't been occupied which results in a new path (x_1, x_6, x_5, x_4).

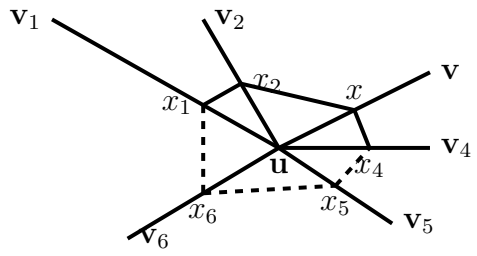


Figure 9: Movement of interior junction

Case 2. $x = (\mathbf{u}, \mathbf{v}, \theta = 1)$ is an exterior junction. Let z be its neighbor on the same obstacle and y be the neighbor on another obstacle. x will move to a different feasible edge **uw** once it hits **u**. Edge **uw** is said to be feasible if

(a) The line segment joining y and $\mathbf{u} + \Delta\theta(\mathbf{w} - \mathbf{u})$ doesn't intersect with any obstacle for arbitrarily small $\Delta\theta$.

(b) The length $J(x)$ decreases as x moves away from **u** on **uw**, i.e.

$$\nabla J(x) \cdot (\mathbf{w} - \mathbf{u}) < 0.$$

We collect all the feasible directions and select one of them with equal possibility, x then continues evolving according to the flow. Depending on whether neighbor z is visible to edge **uw**, the new path are as follows:

i. z is on the same face as **uw**, then the new path becomes $(\cdots, y, x', z, \cdots)$ where $x' \in \mathbf{uw}$.

ii. z is not on the same face as **uw**, then x is used as an intermittent junction and the new path becomes $(\cdots, y, x', x, z, \cdots)$ where $x' \in \mathbf{uw}$.

For an illustration, see the following example. The feasible direction are $\mathbf{uv}_8$ and $\mathbf{uv}_2$. z is visible to $\mathbf{uv}_8$, the path after evolution is simply (y, x', z). On the other hand, z is invisible to $\mathbf{uv}_2$, the path after evolution is simply (y, x', x, z).

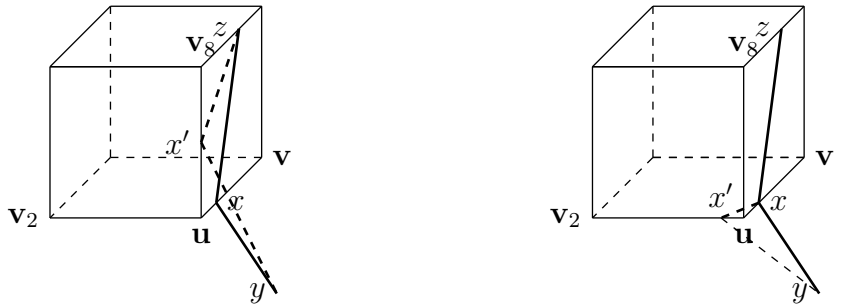


Figure 10: Movement of exterior junction x. The left figure corresponds to case (i) and the right corresponds to case (ii)

Add and remove junctions During the evolution of each point, we may need to add or eliminate junction points. When two neighboring junctions x, y are both exterior and $\overline{xy}$ intersect with obstacle $P_{k_1}, P_{k_2}, \cdots, P_{k_r}$, we initialize a path with x, y being the starting and ending points and $\{P_{k_i}\}_{i=1}^{r}$ being the obstacles. Denote the new added junctions by $(x_{n+1}, x_{n+2}, \cdots, x_{n+s})$ where s is the total number of new junctions. Then they are inserted into the set of junctions in order and the evolution process continues. On the other hand, when two neighboring junctions x, y are both exterior and x meets y, we may shorten the path by removing x and y. More precisely, let z_1 be the other neighbor of x and z_2 be the other neighbor of y, i.e. the path contains $(\cdots, z_1, x, y, z_2, \cdots)$ as a fraction. Since $x = y$, we may connect z_1 and z_2 directly which shorten the length. In other words, we have the new fraction $(\cdots, z_1, z_2, \cdots)$. Notice, the line segment $\overline{z_1 z_2}$ may intersect with some obstacles. Again we add the necessary junctions as described above.

4.5.2.1 Algorithm

We present our algorithm below

Shortest Path amid Polyhedra Obstacles

Input: coordinates of vertices of N polyhedral obstacles,

starting and ending points x and y,

number of intermittent diffusion intervals m.

Output: The optimal set γ_{opt} of junctions.

---

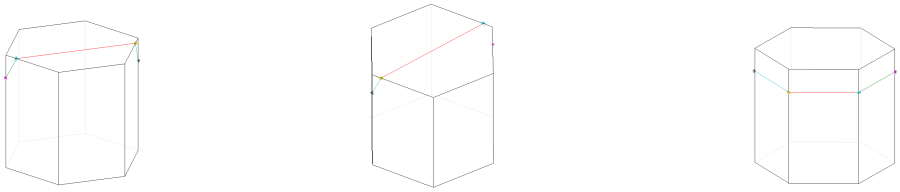
1. Initialization. Find the initial path $\gamma^{(0)} = (x_0, \cdots, x_{n+1})$;
2. Select duration of diffusion $\Delta T_l, l \leq m$;
3. Select diffusion coefficients $\sigma_l, l \leq m$;
4. **for** $l = 1 : m$
5. $\quad \gamma^{(l)} = \gamma^{(0)}$;
6. $\quad$ **for** $j = 1 : \Delta T_l$
7. $\quad\quad$ **for** $x_i = (\mathbf{u}, \mathbf{v}, \theta_i^0) \in U_l$
8. $\quad\quad\quad$ Update x_i according to (145), i.e. $\theta_i^{j+1} = \theta_i^j + (-\nabla J(\theta_i^j) + \sigma_l \sqrt{\Delta t}\xi)\Delta t$;
9. $\quad\quad\quad$ Remove junctions from or add junctions to $\gamma^{(l)}$ when necessary;
10. $\quad\quad$ **end**
11. $\quad$ **end**
12. $\quad$ **while** $|\nabla J(\theta)| > \epsilon$ (or other convergence criterion)
13. $\quad\quad$ **for** $x_i = (\mathbf{u}, \mathbf{v}, \theta_i^{\Delta T_l})$
14. $\quad\quad\quad$ Update x_i according to (145) with $\sigma = 0$, i.e. $\theta_i^{j+1} = \theta_i^j - \nabla J(\theta_i^j)\Delta t$;
15. $\quad\quad\quad$ Remove junctions from or add junctions to $\gamma^{(l)}$ when necessary;
16. $\quad\quad$ **end**
17. $\quad$ **end**
18. Compare $J(\gamma^{(l)}), l \leq m$ and set $\gamma_{opt} = \text{argmin}_{l \leq m} J(\gamma^{(l)})$;

4.5.3 Numerical Examples

We show several examples in this section to illustrate the paths obtained by our algorithm. The diffusion coefficients are chosen randomly in interval $[1, 2]$ and the duration of diffusion ΔT_l is chosen randomly in $[5, 20]$. The number of intermittent diffusion intervals m are specified in each example.

Example 1

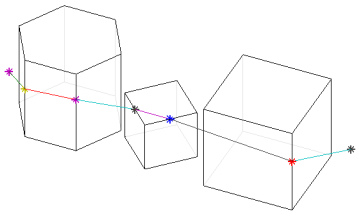
The first example computes the shortest path between two points on a hexagonal prism with side length $\sqrt{3}$ and base length 1. In one realization with $m = 10$ intermittent diffusion intervals, it finds 3 minimizers among which the global one is visited 6 times.



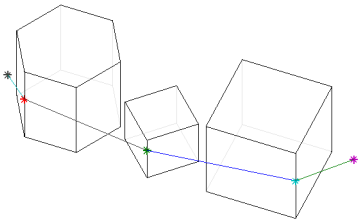
(a) Occurs 6 times, L=2.600 (b) Occurs 3 times, L=2.623 (c) Occurs once, L=3.000

Example 2

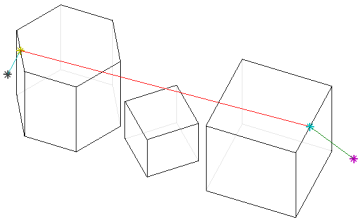
There are three obstacles in this example, two cubes and one hexagonal prism. The algorithm finds 6 local optimal paths in 20 intermittent diffusion intervals, among which the global optimal path occurs 13 times. Below we list all the local minimizers.



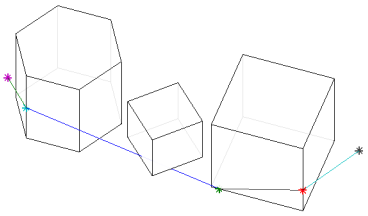
(d) Occurs 13 times, L=7.0803



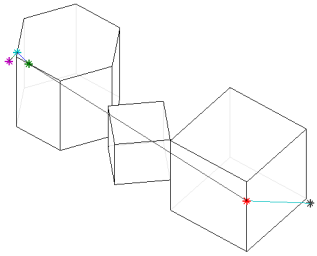
(e) Occurs 2 times, L=7.0956



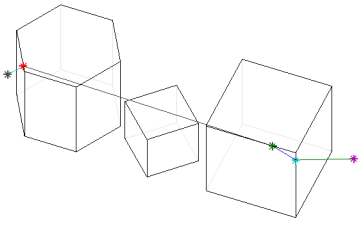
(f) Occurs 2 times, L=7.3404



(g) Occurs 1 times, L=7.3436



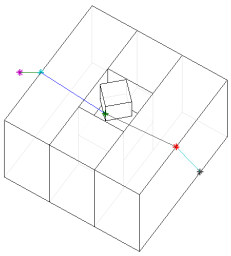
(h) Occurs 1 times, L=7.4314



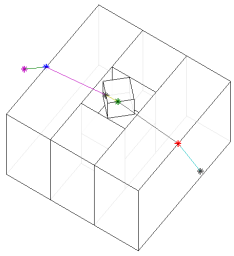
(i) Occurs 1 times, L=7.5253

Example 3

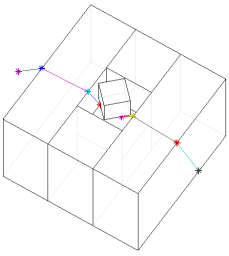
In this example, we demonstrate that our algorithm works for non-convex obstacle without holes. One obstacle is a rotated cube and the other one is a larger cube with a indentation (not through). In 20 intermittent diffusion intervals, the algorithm finds 4 local optimal path. The global shortest path is visited 14 times.



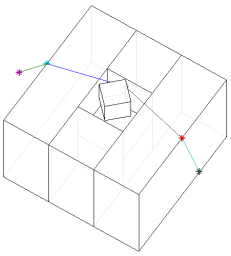
(j) Occurs 14 times, L=4.4249



(k) Occurs 2 times, L=4.4599



(l) Occurs 3 times, L=4.6176



(m) Occurs 1 times, L=4.5509

4.5.4 Polyhedron with Holes

We say two paths are homotopic if one can be deformed continuously to the other while keeping its endpoints fixed. More precisely, let $\mathcal{X}$ be the space that takes away

all the obstacles, i.e.

$$\mathcal{X} = \mathbf{R}^3 \setminus \bigcup P_i.$$

Two paths f_0, f_1 are path-homotopic if there exists a family of paths $f_t \colon [0,1] \to \mathcal{X}$ such that

1. $f_t(0) = x_0$ and $f_t(1) = x_1$ are fixed.

2. the map $F \colon [0,1] \times [0,1] \to \mathcal{X}$ given by $F(s,t) = f_t(s)$ is continuous.

Intuitively, two paths are homotopic if one can be continuously transformed to the other without passing through the obstacles. Path-homotopic is a equivalence relation. Thus one can divide all path into equivalence classes. It is easy to see the following

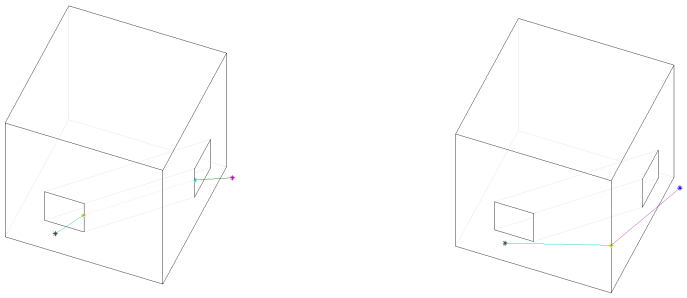
Theorem 22 *If all the obstacles have Euler characteristic 2, then there is only one path-homotopic equivalence class in the set of feasible paths F.*

Proof 7 *Since each P_i has Euler characteristic 2, P_i is homotopy to 2-dimensional sphere S^2. Notice that $R^3 - D^2$ where D^2 is the 2-dimensional disk is trivial. Therefore, any two path in $R^3 - D^2$ are path-homotopic. Same result holds for R^3 taking away n disks.*

With this result, it is simple to obtain the results in Theorem 157.

Proof 8 (Proof of Theorem 157) *Theorem 22 guarantees that our algorithm is able to visit all possible paths from any initialization. Therefore, by employing intermittent diffusion, equation* (145) *converges to the global minimizer with probability $1 - \epsilon$ for suitable $\sigma(t)$.*

However, on the contrary, if the obstacle contains holes, for example, a triangulated torus, there would be multiple equivalence classes. For illustration, see the following tunneled cube in Figure (11). The shortest path through the hole is 1.1543



(n) L=1.1543 (o) L=1.2659

Figure 11: Shortest path with tunneled cube

while the one doesn't has length 1.2659. By slightly changing the position of the hole, the shortest path would be the one that not pass through it. Therefore, multiple initializations are needed to ensure that all possible equivalence classes are covered.

A simple idea we can use is to block the homotopy equivalence class the current path belongs and then reinitialize. A block can be formed ,for example, by identifying the entrance of the path followed by deleting all the vertices on it. Those vertices will not be used in the reinitialization which forces the new path to a different homotopy class. After it settles down at the global minimizer in the current homotopy class, the path is reinitialized and the algorithm is repeated to get a different global minimizer. This procedure is repeated until all homotopy equivalence classes are visited. The two paths in the above example are obtained in this method.

However, there are two problems with this approach. First of all, the block is often difficult to form because which vertices should be removed is a complicated matter, for instance, a well triangulated torus as follows. Second, the number of different homotopy classes we need to visit is unknown in prior. For example, topologically, there are infinitely many homotopy classes for a smooth torus and the shortest path could wind the torus arbitrary times. In Figure (13), the shortest path winds the torus twice.

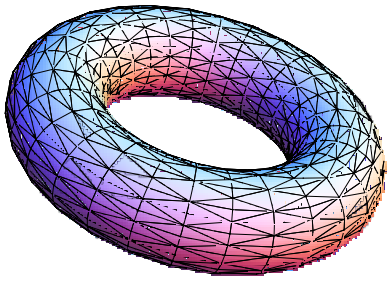


Figure 12: A triangulated torus

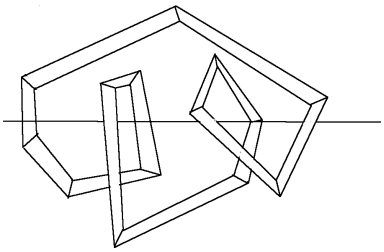


Figure 13: A shortest path winding a torus twice

A different approach is to use already established approximation method, for example [12] as described in the introduction section, to initialize the path. Those algorithms are able to obtain a path that has length $1 + \epsilon$ times the length of the shortest path. Here ϵ depends on the mesh size. If the mesh size is sufficiently small, the initialized path and the global minimizer will be in the same homotopy class. However, the choice of the grid size is a critical and often hard to determine.

As discussed above, our method still applies for polyhedrons with holes provided that appropriate initializations are taken. Although initialization is a complicated matter, simple ideas usually work for most cases. We conclude our discussion here and leave the improvement of initialization methods to our future work.

4.6 Some Extensions

4.6.1 Shortest Path for Moving a Disk

In this section, we consider the shortest path between two points for a disk D with radius r. This is a more realistic model in robotics where the moving system, such as the Unmanned Vehicle Systems (UVS), is a rigid body free only to translation. The distance is defined to be the total distance traversed by the center of the moving system. Given the size of the mover, the feasible path for a point may become infeasible for the disk. To cope the scenario, we take the idea of replacing the obstacles, denoted their region by P, by their expanded versions to account for the size of the disk. One way to achieve this is to form the Minkowski sums $\bar{P} = P \oplus (-D)$. In the level set framework, this can be accomplished easily. Suppose $\phi_k(x)$ is the signed distance function that represents the obstacle, then $\phi_k(x) - r$ represents the enlarged obstacle by length r as desired. In this case, A path is said to be feasible if the disk centered at $\gamma(t)$ doesn't intersect with any obstacle, i.e.

$$\phi_k(\gamma(t)) > r, \quad t \in [0,1], \quad 1 \le k \le N$$

Let us define $\bar{\phi_k} = \phi_k - r$, then it is easy to see that

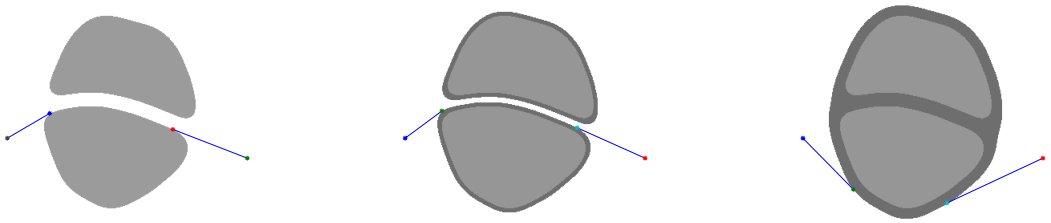
Theorem 23 *A path $\gamma(t)$ for the disk D is feasible if and only if*

$$\bar{\phi_k}(\gamma(t)) > 0, \quad t \in [0,1], \quad 1 \le k \le N.$$

The length of the path is naturally taken to be the distance that the disk center traverses. Thus the shortest path for a disk is equivalent to the shortest path for a point in the new environment where level set function ϕ_k for the original obstacles are replaced by $\bar{\phi_k}$. With this setup, the path can be readily computed by E-JOB. The following is an example for moving a disk.

Example. In Figure (14), we illustrate how radius of the disk affects the shortest path. The left picture shows the shortest path for a point mover obtained by E-JOB.

It goes through the tunnel. The middle picture is the path for a disk with radius $r = 0.02$. The obstacles are flattened. The darker area is the new region in the enlarged obstacles represented. The tunnel is narrowed. Since the radius of the disk is relatively small, the tunnel is large enough for the disk to pass through. However, when the radius increases up to 0.05, the tunnel is stuck which forces the shortest path to go "outside" of the obstacle (shown on the right of Figure 14).



(a) $L = 0.6266$, $r = 0$ (b) $L = 0.6347$, $r = 0.02$ (c) $L = 0.6952$, $r = 0.05$

Figure 14: Example: shortest path with disk movers: Left: a point mover, the shortest path is through the tunnel. Middle: a disk with radius $r = 0.02$. The tunnel is still large enough for the disk to pass through. Right: A disk with radius $r = 0.05$, which is too large to move through the tunnel. The shortest path must go outside.

4.6.2 Shortest Path Between Two Sets

In the second problem, we consider the Euclidean distance between two sets P and Q in $\mathbf{R}^n$, i.e. the shortest path between them. This is an important problem in computer-aided design and computer graphics in which people often want to know whether two objects intersect or how far there are. In [26], the authors focus only on convex sets. Later, another method is proposed to deal with non-convex objects by breaking them into convex components and then apply the algorithms for convex sets to determine the distance between components [55]. The authors also reduced the complexity significantly by eliminating the number of possible pairs of components. In this paper, we demonstrate that the method of evolving junctions can be easily modified to solve this problem. The idea is to treat the starting point X and the

ending point$Y$ as undetermined junctions and evolved on the boundaries of P and Q respectively. An clear advantage of E-JOB is that it places no restrictions on the obstacles, whether they are convex or non-convex, discrete or having smooth boundaries. We consider in this section the shortest path problem when the starting and ending points X, Y become two sets P and Q, in other words, the distance between two Euclidean regions. If P and Q are treated as two new obstacles, the problem differs little to the original shortest path problem in the framework of E-JOB. In fact, the length of the two segments adjacent to junction x_i, $J(x_i)$, remains the same while x_0 and x_{n+1} now become two new undetermined junctions instead of being fixed. Therefore, the length functional is

$$L(\gamma) = J(x_0, x_1, \ldots, x_n, x_{n+1}) = \frac{1}{2}\sum_{i=0}^{n+1} J(x_i).$$

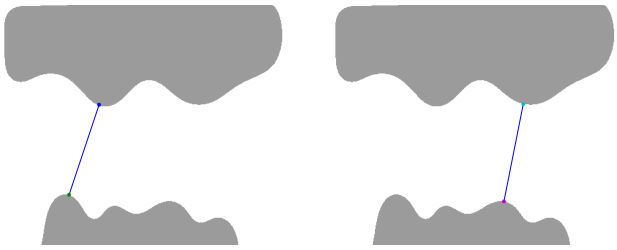
Theorem 24 *Theorem 157 still holds for the shortest path problem between two sets, except that line segments $\overline{x_0x_1}$ and $\overline{x_{n+1}x_n}$ are perpendicular instead of being tangent to the boundaries.*

Proof 9 *The first part follows exactly as in the proof in [17]. For x_0, its critical point satisfies*

$$\nabla J(x_0) = \frac{x_0 - x_1}{||x_0 - x_1||} \cdot \mathbf{T} = 0 \tag{146}$$

Hence, $x_0 - x_1$ is perpendicular to $\mathbf{T}$, i.e. the boundary of the obstacle x_0 is on.

Unlike the initialization as stated in the algorithm in Section 4.4, it is more challenging to find a good initialization here. The reason is that the starting point x_0 and ending point x_{n+1} are variables themselves. When X and Y are fixed, the initialization which consists of the intersections of the line segment $\overline{XY}$ and all the obstacles is effective and efficient in the sense that it leads to the globally shortest path more quickly. However, when X and Y varies, there is no particular criterion to tell which initialization is better than the other. As a result, the initialization we choose is



(a) $L = 0.2840$ (b) $L = 0.2979$

Figure 15: Shortest path between two sets. The left one is the global optimal solution.

simply randomly select x_0 and x_{n+1} on the boundaries of P and Q respectively. For the same reason, it usually takes more time to converge to the global optimizer. We illustrate it by the following example,

Example We compute the distance between two islands whose boundaries have a couple of bumps. It is not hard to see there are at least 16 local minimizers. We show two of them in Figure 15. Among them, the left one is the globally shortest path.

4.6.3 Shortest Path with Obstacles Appearing or Disappearing

The last problem concerns the shortest path in an environment that the obstacles appear or disappear over time. It is not our intention to investigate the time-varying shortest path problem as described in [38] which studies how the robot moves to reach Y with the minimal cost in a dynamic environment, which includes but not limited to moving, shape changing(shrinking, expanding, etc) or appearing/disappearing obstacles. Instead, we focus on how to recompute the shortest path with appearing or disappearing obstacles assuming that E-JOB runs in real-time. This is an extremely challenging situation for the existing methods. For example, in the method of front propagation, the distance function at all grid points must be recomputed to find the solution once an obstacle appears or disappears. This could be very costly due to the involvement of solving PDEs multiple times. However, E-JOB can treat it efficiently, because many junctions remain unchanged, although some of them appear

or disappear as the appearance or disappearance of obstacles. This means the shortest path stays the same mostly regardless of the changing environment. Since all of the remaining junctions already settle down, only local updates of the junctions are required .

This section concerns the shortest path between two fixed points in a dynamic environment. More specifically, we consider the scenario in which obstacles appear or disappear. We would like to stress that it is not the usual time-varying shortest path problem as discussed, for example, in [38]. The main difference is that the functional (length of the path) we want to minimize doesn't involve time because the obstacles are not in motion. This is still a common scenario in practice, especially in robotics. And E-JOB provides an extremely low-cost solution as compared to other existing algorithms, such as the PDE based methods. In fact, as we demonstrate in the examples, it often requires only local updates to obtain the shortest path by E-JOB with appearing or disappearing obstacles.

For simplicity, we only consider the scenario in which one obstacle pops out or disappear. The case in which multiple obstacles appear or disappear can be handled in the same manner. Suppose we have N obstacles initially and the shortest path between X and Y is obtained by E-JOB with the following junctions,

$$\gamma = (x_0, x_1, \ldots, x_n, x_{n+1}).$$

Assume that a new obstacle, denoted by P_{N+1}, appears. Without loss of generality, let us assume that P_{N+1} only intersects with line segment $\overline{x_k x_{k+1}}$ for some k and denote the intersections by

$$(x_k, x_{n+2}, x_{n+3}, \ldots, x_{n+l}, x_{k+1}). \tag{147}$$

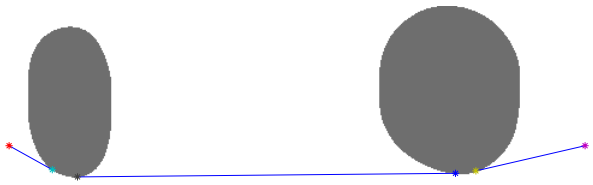
The new set contains $n + l + 1$ junctions in total. One can apply E-JOB to obtain the globally optimal solution if needed. However, if a local minimizer is needed, only $l + 1$ junctions need to be evolved. The junctions not in (147) are already settled

down. Moreover, from our experience and the examples showing in this section, since $x_0, \ldots, x_{n+1}$ is part of the globally optimal path, the local minimizer obtained by evolving $x_{n+2}, \ldots, x_{n+l}$ is very likely to be the global minimizer, although this is not necessarily always the case.

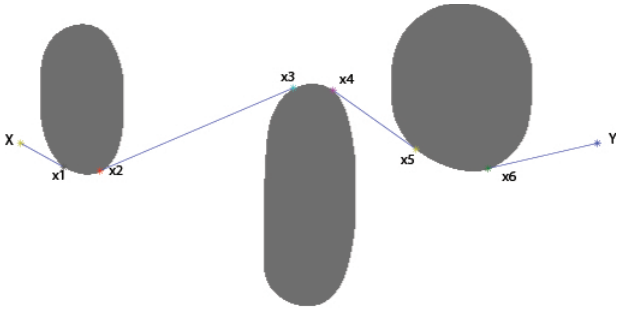
The case in which an obstacle disappears can be handled in a similar manner. Assume obstacle P_k disappears at certain time. If there were no junctions on the boundary of P_k, the shortest path remains the same. If x_i and x_{i+1} were on P_k, we shorten the length by directly connecting x_i^s and x_{i+1}^s with a line segment and removing junctions x_i and x_{i+1}. If line segment $\overline{x_i^s x_{i+1}^s}$ intersects with any existing obstacles, we add the intersections as new junctions.

Examples. We give four examples with two showing obstacle appearing and two for obstacle disappearing . In the first two examples (Figure 16a,17a), there are two obstacles initially. At certain time, a new obstacle (in the middle) pops out and changes the shortest path accordingly. In the first example, the local minimizer (16b) obtained by evolving new junctions x_3, x_4 is the global optimizer. Notice that x_1, x_6 remain the same. On the other hand, the second example demonstrates the contrary situation. Local updates in Figure (17b) only leads to a local minimizer. All the junctions x need to be recomputed in order to obtain the globally shortest path (17c).

The case in which the middle obstacle disappears are represented by the same figures with reversed orders. More specifically, Figure (16b,17c) are new initial environments and Figure (16a,17a) are the globally shortest path respectively after the middle obstacle disappears.

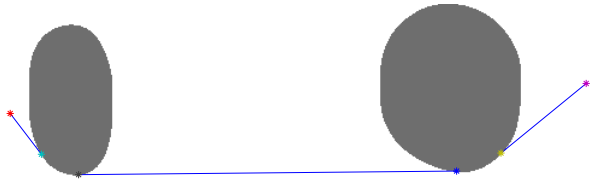


(a) Initial environment, shortest path has length $L = 0.9400$

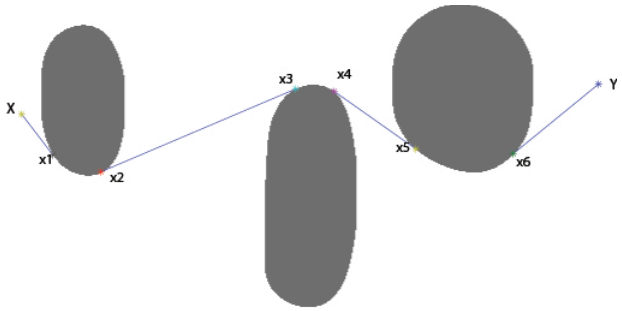


(b) One obstacle appears, globally minimizer has length $L = 0.9992$

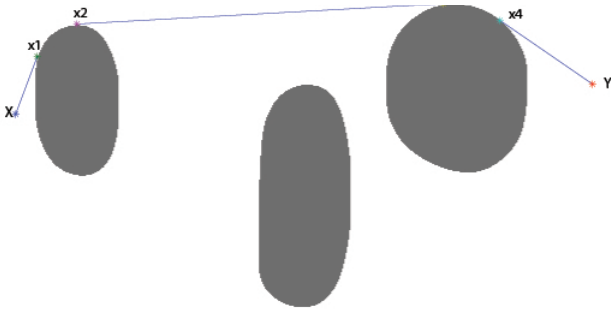
Figure 16: Shortest path with obstacle appearing (Example 1)



(a) Inital environemnt, shortest path has length $L = 1.0211$



(b) One obstacle appears, local minimizer has length $L = 1.0784$



(c) One obstacle appears, globally minimizer has length $L = 1.0593$

Figure 17: shortest path with obstacle appearing (Example 2)

CHAPTER V

APPLICATIONS TO ROBOTICS

5.1 Introduction

In this chapter, we will apply the algorithm we developed previously to find the optimal path for a UAV in a cluttered environment. Path planning is one of the enabling technologies making it possible for robots to successfully traverse cluttered environments and, as such, it has received considerable attention in the robotics community. (See for example the textbooks [38, 37, 24] and the references therein for a comprehensive treatment on robotic path planning.)

Since globally optimal path planning is hard, the computational solutions that have been proposed typically fall into three different camps, namely (i) grid-based planners, such as A* [29] and D* [66, 67, 36] , (ii) sample-based planners, such as RRTs [39] or (iii) continuous deformations of locally optimal plans, e.g., [56, 42, 8]. This does not mean that global planners have not been developed, as was done for example in [8, 17, 61], but what it means is that it is in general quite hard to transition global planners from an off-line setting to an on-line robotics implementation.

Here we focus on the problem of making an aerial robot traverse an environment populated by obstacles following the shortest path. Optimal planning for aerial vehicles maneuvering environments with obstacles where multiple waypoints must be achieved has been studied with respect to the vehicle dynamics and other tactical constraints [57, 73], however this often results in local minimizers or it is constraint to two dimensions. In addition, many existing planning algorithms either focus on finding a feasible path (collision-free) [42] or optimizing a different objective function rather than the length of the path [8, 48]. In fact, finding the globally shortest path

has been proven to be NP-hard in 3-D with polyhedral obstacles in the framework known as configuration space by Canny and Rief [7]. Approximation methods, which are the only practical approaches, seek paths whose length is $1+\epsilon$ times the actual minimal path [13, 14, 53, 33]. However, those methods are either only theoretical and only works for polyhedral obstacles, or have high complexity which are unsuitable for on-line implementation. In this paper, we will take advantage of the recently developed method in [17], where intermittent diffusion techniques are employed to get increasingly good (short) paths in arbitrary dimensions. The resulting algorithm has the following advantages over existing methods: (i) it is able to find the globally shortest path with probability $1-\delta$ with complexity $O(\log\frac{1}{\delta}\log\frac{1}{\epsilon})$ for arbitrarily small δ, to the best of our knowledge, this is the best complexity occurred in the literature; (ii) the main computation in the algorithm lies in solving initial value ODE's or stochastic ODE's, which is very easy to implement; (iii) the algorithm can be adapted to handle dynamic environments, in which some obstacles may appear or disappear. These three features will allow us to have an aerial quadrotor negotiate a cluttered environment effectively and the main contribution of the paper should be understood in terms of solving the 3-D shortest path problem in an implementable, on-line manner, that is actually deployed on a real robotic vehicle.

5.2 Planning Approach

Consider N obstacles in $\mathbf{R}^3$, each of which is represented by a connected and compact set $P_k \subset \mathbf{R}^3$ $(1 \le k \le N)$. We assume that all of the obstacles are pairwise disjoint and their boundaries have certain regularities, for example being piecewise C^2. For convenience of computation, we will represent each obstacle through the signed distance function $\phi_k(x)$,

$$\phi_k(x) = \begin{cases} -p_k(x), & x \in P_k \\ p_k(x), & x \in P_k^c \end{cases} \tag{148}$$

Figure 18: A Parrot AR.Drone quadrotor robot.

where P_k^c denotes the complement of P_k and $p_k(x)$ is the Euclidean distance from x to ∂P_k (the boundary of P_k),

$$p_k(x) = \min_{y \in \partial P_k} \|x - y\|.$$

The signed distance function can be computed efficiently through level set methods [51].

For any two points $x, y \in \partial P_k$, let $J_k(x, y)$ be the length of the shortest geodesic (shortest path) connecting x and y that remains entirely on the boundary. For special shapes of obstacles such as cubes, cylinders and cones, $J_k(x, y)$ can be computed analytically. For polyhedral obstacles, $J_k(x, y)$ can be computed efficiently by the algorithms in [60], or see Remark 5 for more information. The general, more complicated cases can be handled by fast marching [63] or fast sweeping [75], or approximated by polyhedrons, with a trade off between accuracy and complexity.

In this paper, the planner will assume that $J_k(x, y)$ is given a priori. The environment, however, is allowed to be adaptive, in the sense that in the experimental

implementation of the planning algorithm, the environmental knowledge will be incrementally updated.

The path we are interested in is a curve γ in $\mathbf{R}^3$, which is a continuous map $\gamma\colon [0,1] \to \mathbf{R}^3$. We let $J(\gamma)$ be the length of γ under the Euclidean metric.

A path with finite length is said to be rectifiable, and we say a path is feasible, in a geometric sense, if γ does not intersect with the interior of any obstacles. Formally, we require

$$\phi_k(\gamma(t)) \geq 0, \quad t \in [0,1], \quad 1 \leq k \leq N. \tag{149}$$

As the quadrotors have a spatial footprint, we need to be able to accommodate this situation as well. The problem of how to consider the geometry of the robot is out of the scope of this paper and has been addressed before [38]. However, since the spatial footprint of the quadrotor is quite symmetrical, at least along certain dimensions, the robot's footprint can be approximated by a ball of radius r, such that the robot can be treated as a ball robot.

The level set representation of obstacles in (148) enables us to deal with ball robots in a straightforward manner. One only needs to enlarge all obstacles by this radius r, which boils down to a uniform decrease in the level set function. We may restate the definition of feasibility to: a path is said to be feasible for a ball robot with radius r if

$$\phi_k(\gamma(t)) - r \geq 0, \quad t \in [0,1], \quad 1 \leq k \leq N. \tag{150}$$

It is noteworthy that this approach of wrapping the robot inside a ball can be done even when the robot's footprint is not symmetrical. Even though it is possible to produce better approximations for a given robot footprint, we use this approximation for convenience.

Let F be the set of all feasible rectifiable paths for a ball robot with radius r which starts from X and ends at Y ($\gamma(0) = X, \gamma(1) = Y$). Our problem is to find a path in

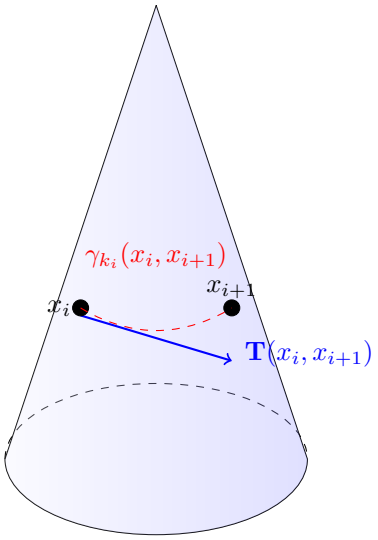


Figure 19: An illustration of $\mathbf{T}(x_i, x_i^c)$.

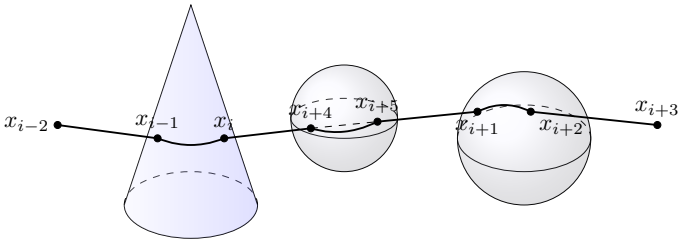


Figure 20: The initial path is $(\cdots, x_{i+2}, x_{i+3}, \cdots)$. As the path changes, $\overline{x_{i+2}x_{i+3}}$ intersects with the obstacle in between. We add the intersections points as junctions and the new path is $(\cdots, x_{i+2}, x_{i+6}, x_{i+7}, x_{i+3}, \cdots)$.

F such that its length is minimized:

$$\gamma_{opt} = \underset{\gamma \in F}{\operatorname{argmin}}\ J(\gamma). \tag{151}$$

On the other hand, when two straight components $\overline{x_i z}, \overline{x_j z}$ share a common junction z, the path can be shortened by connecting $x_i x_j$ directly, as long as this straight line does not intersect another obstacle. We hence remove z from the set of junctions. If $\overline{x_i x_j}$ does intersect another obstacle, then new junctions must be added as previously described. See Fig. 21.

Remark 4 (Dynamic Environments) *The approach described above can also be used when the environment is not fixed but changes over time. When obstacles appear, it is only necessary to introduce the new junctions as described above if an obstacle*

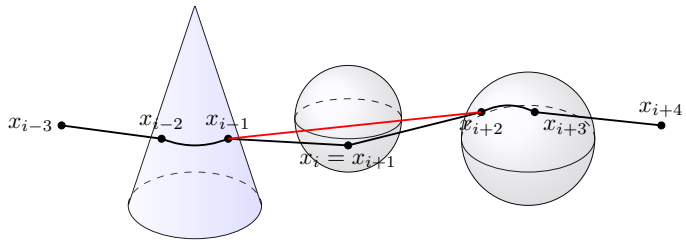


Figure 21: The initial path is $(\cdots, x_{i+2}, x_{i+3}, x_{i+4}, x_{i+5}, \cdots)$. At some time during the path change, $x_{i+3} = x_{i+4}$. The path can be shortened by connecting x_{i+2} and x_{i+5}. The new path becomes $(\cdots, x_{i+2}, x_{i+5}, x_{i+6}, \cdots)$.

instersects with the current path. Likewise, when obstacles disappear, the path can be shortened as described above.

Remark 5 (Polyhedral Obstacles) *For polyhedral obstacles, the proposed method can be equally applied to computing the geodesics on the surface. More specifically, for any two points $x, y \in P_k$, there exists a sequence of junctions*

$$(z_0, z_1, \cdots, z_l), \quad z_0 = x, \quad z_l = y, \tag{152}$$

such that each z_i is on some edge of P_k and

$$J_k(x, y) = \sum_{i=0}^{l-1} \|z_i - z_{i+1}\|. \tag{153}$$

To determine the z_i's, it suffices to optimize (153) *over all possible* (152)*. For more details, especially how to handle the topological changes of the path during evolution, we refer readers to [12].*

5.3 Robotic Implementation

In this section an algorithm is proposed to find the set of junctions that represents the shortest path between two points in an environment with obstacles. The feasibility, in a dynamical sense, of the algorithm is then demonstrated when it is implemented on a Parrot AR.Drone quadrotor robot, seen on Fig. 23 and Fig. 24.

5.3.1 Algorithm

The following algorithm is the same as 4.4.1.3. We quote it here in order to analyze its complexity.

Shortest Path in 3-D Cluttered Environment

Input: level set function $\phi_k(x)$,

the distance function $J_k(x, y)$,

starting and ending points X and Y,

number of intermittent diffusion intervals m.

Output: The shortest path γ_{opt} represented by a set of junctions.

---

1. Initialization. Find the initial path $\gamma^{(0)} = (x_0, \cdots, x_{n+1})$ which consists of all the intersection points $\overline{XY}$ with the boundary of the obstacles;
2. Select duration of diffusion $\Delta T_l, l \leq m$;
3. Select diffusion coefficients $\sigma_l, l \leq m$;
4. **for** $l = 1 : m$
5. $\quad \gamma^{(l)} = \gamma^{(0)}$;
6. $\quad$ **for** $j = 1 : \Delta T_l$
7. $\quad\quad$ Update $\gamma^{(l)}$ according to (131) with $\sigma(t) = \sigma_l$;
8. $\quad\quad$ Remove junctions from or add junctions to $\gamma^{(l)}$ when necessary;
9. $\quad$ **end**
10. $\quad$ **while** $\|\nabla^c J(\gamma^{(l)})\| > \epsilon$
11. $\quad\quad$ Update $\gamma^{(l)}$ according to (127);
12. $\quad\quad$ Remove junctions from or add junctions to $\gamma^{(l)}$ when necessary;
13. $\quad$ **end**

14. **end**

15. Compare $J(\gamma^{(l)}), l \leq m$ and set $\gamma_{opt} = \text{argmin}_{l \leq m} J(\gamma^{(l)})$;

The initial set γ^0 of junctions consists all the intersections points of line segment $\overline{XY}$ and the obstacles. This initialization gives initial path similar to those generated by bug algorithms [42], and in many cases is already close enough to the global optimizer. It should be noted that good initialization is not required for the proposed algorithm. Given enough time to run the algorithm, the global minimizer will still be obtained even starting from a far initial path.

Both the duration of diffusion ΔT_l and diffusion coefficients σ_l are randomly selected in intervals $[0, T_{\max}]$ and $[0, \sigma_{\max}]$ respectively. Experiments show that $T_{\max} = 20$ and $\sigma_{\max} = 2$ are often adequate. Depending on whether one wants to record local minimizers, line 20 can be replaced by keeping track only of the best minimizers at current realization. This will save storage dramatically.

We now give a brief analysis of the algorithm

5.3.1.1 Completeness

Since we assume all the obstacles are bounded and disjoint, and we start from a feasible path, Theorem 7 guarantees the proposed algorithm is complete.

5.3.1.2 Complexity

Following [53], instead of discussing the algebraic complexity of the algorithm, we will consider the running time in order to achieve certain relative error ϵ. We will compare our result with other approaches only for polyhedral obstacles since most of the literature focus on them.

1. The initialization can be done by a bisection line search, which can be achieved in $O(\log \frac{1}{\epsilon})$ time..

Table 2: Comparison of complexty of different algorithms.

Algorithm	Complexity
A^*	$O((\frac{1}{\epsilon})^3 \log \frac{1}{\epsilon})$
Papadimitriou [52]	$O(\frac{1}{\epsilon})$
Choi, et. al [12]	$O(\frac{1}{\epsilon})$
Choi, et. al [12] When the shortest path is unique	$O(\log \frac{1}{\epsilon})$

2. Line 2-3 takes $O(m)$ time.

3. Inner loop line 6-9 takes $O(\Delta T_l)$ time. This is because equation (131) takes constant time, and so does adding or removing junctions.

4. Inner loop line 10-13 takes $T(\epsilon)$ time where $T(\epsilon)$ denotes the number of iterations required until the error is less than ϵ. If we assume the Hessian matrix of the gradient is nondegenerate, which is the case for all polyhedral obstacles [13], then $T(\epsilon) = O(\log \frac{1}{\epsilon})$.

Let $\Delta T = \max_{i \leq l} \Delta T_i$. Then the total running time is $O(m(\Delta T + \log \frac{1}{\epsilon}))$. From [16], it can be shown that in order to obtain the desired successful probability $1 - \delta$, the number of realizations must be of order $O(\log \frac{1}{\delta})$. Therefore, the complexity is $O(\log \frac{1}{\delta} \log \frac{1}{\epsilon})$. The following table shows a complexity comparison with some existing methods.

Table 3: Improvement on Probability of Obtaining Shortest Path, Environment A.

Minimizer	**Length (m)**	**Times Obtained Out of**		
		100	**200**	**300**
1	5.8660	48	103	159
2	5.9527	50	91	128
3	6.0403	1	4	6
4	6.0594	0	1	1
5	6.0919	0	0	1
6	6.2286	0	0	3
7	6.2305	1	1	2

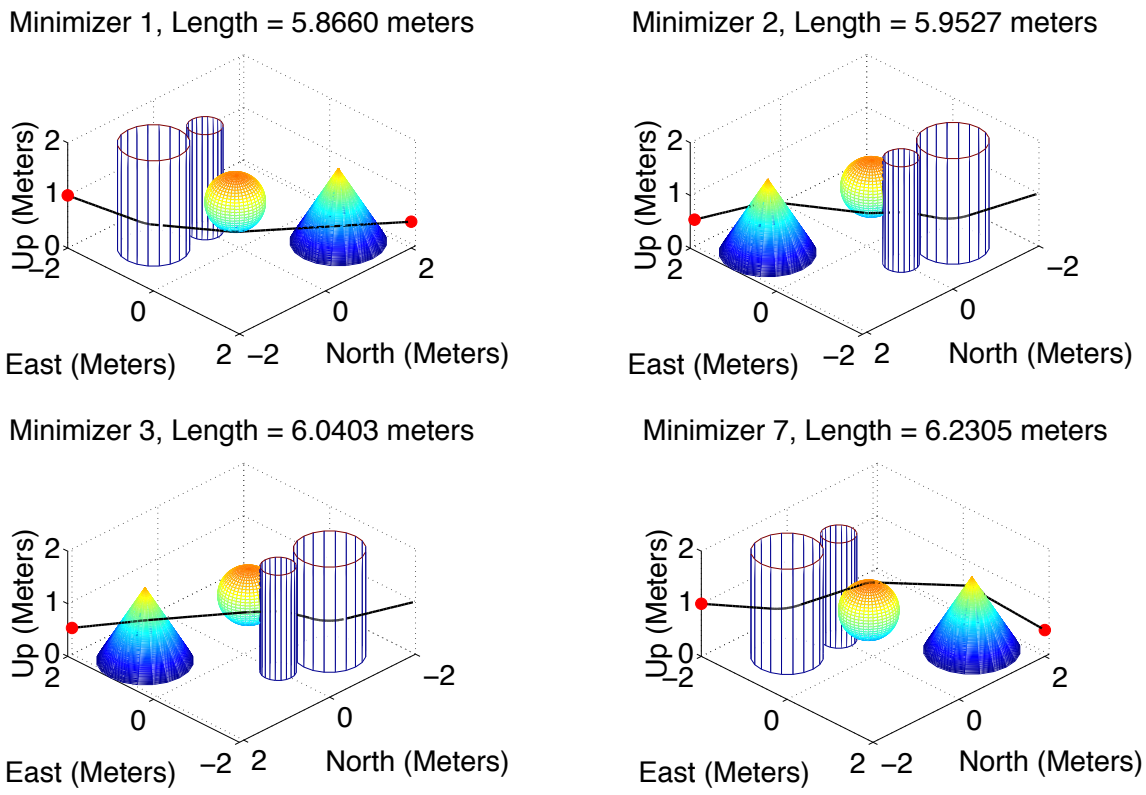
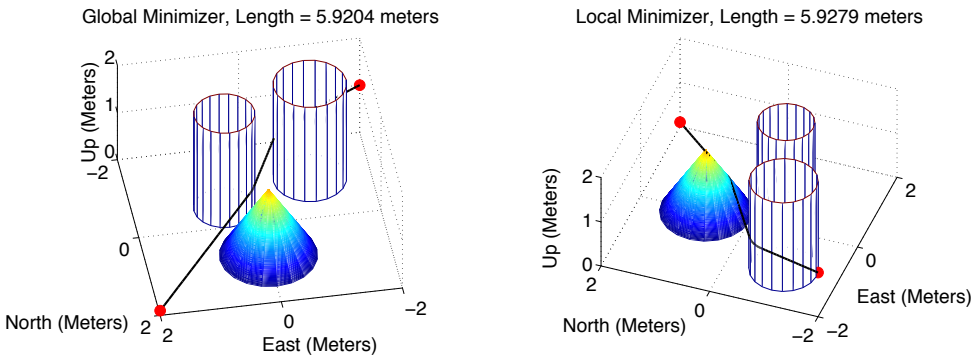


Figure 22: Three shortest and longest minimizers for environment A

5.3.2 Algorithm Implementation

The algorithm described above was implemented for several environment definitions, as seen in Fig. 22 for environment A, and Fig. 23b and Fig. 23a for environment B. Fig. 22 presents several of the local minimizers found by the algorithm for environment A. Table 3 shows how, out of seven local minimizers found in environment A, the number of times the shortest path was found improves as more time is allowed for the algorithm to run, i.e. when more realizations (outer-most loop of the algortihm) are permitted. The sorting effect is also evident where the second and third shortest minimizers also improve in probability of being obtained by the algorithm. Fig. 23b and Fig. 23a are minimizers found for environment B. In this case, the number of minimizers is less than in environment A, however the total distance for each minimizer is much closer together. The algorithm is still able to discern between the local and global minimizer, as can be seen in table 4. It is noteworthy that even



(a) (b)

though these paths are very close to each other in total length, their geometry is very different. Depending on the scope of the application, one may choose to follow a local minimizer rather than the global minimizer, trading off distance for alternate path geometry.

5.3.3 Environment Definition

Environment B is considered for the robotic implementation and experimental setup that will be the focus of the next sections. As seen in Fig. 23b and Fig. 23a, three obstacles are considered, one cone with base center $(0, 0.8, 0)$, base radius $0.8m$ and height $0.8\sqrt{3}m$; one cylinder with base center $(-1, -1, 0)$, radius $0.6m$ and height $2m$; one cylinder with base center $(0.5, -0.5, 0)$, radius $0.5m$ and height $2m$. The starting and ending points are located at $(-2, -2, 1)$ and $(2, 2, 0.1)$ respectively. The proposed algorithm is able to find two minimizers as shown in Fig. 23b and Fig. 23a. Fig. 23a is the globally shortest path whose length is approximately $5.9204m$. This is in contrast to the distance between the initial and final position which is approximately $5.728m$

Table 4: Improvement on Probability of Obtaining Shortest Path, Environment B.

Minimizer	**Length (m)**	**Times Obtained Out of**		
		100	**200**	**300**
1	5.9204	52	104	158
2	5.9279	48	96	142

and the local minimizer length which is approximately $5.9279m$, with a difference of only $0.0075m$ between the local and global minimizer.

5.4 Conclusion

The problem of having a robot traverse a 3-dimensional environment with obstacles was considered. An algorithm was presented that utilizes intermittent diffusion techniques to provide the global distance minimizer, or shortest path, with increasing probability as more time is allowed for the algorithm to run. A series of local minimizers are also provided for situations where more time is not possible. A brief completeness and complexity analysis is provided for the algorithm together with a comparison to other existing algorithms. Furthermore, the presented algorithm is implementable in an on-line manner for robots with the capability of detecting obstacles and approximating complex shapes to composition of simpler shapes. The algorithm is validated when it is successfully implemented in a quadrotor robot.

Figure 23: A third of the way: front, profile and back view.

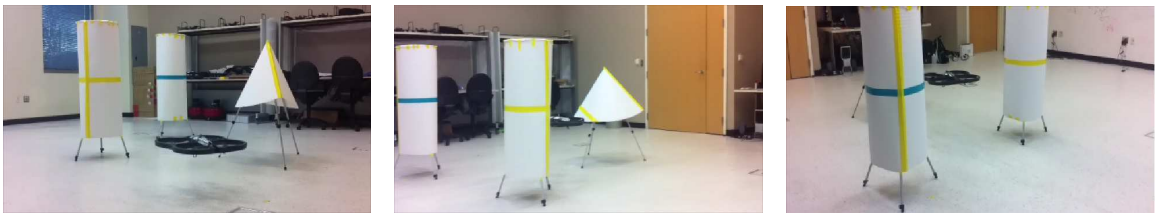


Figure 24: Two thirds of the way: front, profile and back view.

CHAPTER VI

MINIMAL COST PATH IN DYNAMIC ENVIRONMENT

In this chapter, we consider the problem of finding the minimal cost path in a dynamic environment, in which, for example, the obstacle are moving or expanding/shrinking. Most of literature only focus on how to find a collision free path, which already turns out to be nontrivial. For related work, we refer readers to [38]. The objective functional in dynamic environments is more complicated than that in static environments. It can be the distance or time one travels, or some combination of the two. Let $d(t, Y)$ denote the shortest distance from X to Y using time t. Then the shortest distance is $\min_t d(t, Y)$ and the shortest time is $\min\{t \mid d(t, Y) > 0\}$. Function $d(t, Y)$ can be computed by dynamic programming. More specifically, let $d(t, x)$ be the shortest distance from X to x using time t. It can be shown that $d(t, x)$ satisfies

$$\begin{cases} \frac{\partial d}{\partial t} = \min\limits_{\substack{u \in F \\ 0 \le r \le v_m}} r(1 + \nabla d \cdot u), & x \text{ is not occupied at time } t; \\ d(t, x) = \infty, & \text{otherwise.} \end{cases} \tag{154}$$

Here $F \subset S^1$ is the set of feasible unit directions and $u \in F$ is feasible if $x + \Delta x \cdot u$ is not inside any obstacle for small Δx. However, this method is based on solving PDEs, which is not efficient in higher dimensions. In [71], a novel approach was developed for the simple case where the obstacles are expanding disks and the performance index is the time consumed and in [18], the disappearing/appearing case was handled successfully.

The chapter will be arranged in the following way. In section (6.1), we give some necessary mathematical background, mainly on the rigorous definition and properties of paths in a metric space. A more general statement of the problem in the metric space helps identify the key to the separability, which we introduce in section (6.2).

In section (6.2), we show that in $\mathbf{R}^2$, under mild conditions on the performance index and the motion of obstacles, the optimal path is separable. In section (6.3), we apply the method of evolving junctions methods to find the minimal cost paths in an environment with moving obstacles. Section (6.4) shows one example in which there are five moving obstacles.

6.1 Optimal Path amid Dynamic Obstacles

Let (X, d) be a length space and $\{P_1(\theta), P_2(\theta), \cdots, P_N(\theta)\}$ be N time-dependent open subsets of X, representing obstacle, such that the boundary of each obstacle ∂P_k is also a complete locally compact length space with metric d_k. Given two points $x, y \in X_c(\theta) = X \setminus \cup_i^N P_i(\theta)$, define the admissible set of curves to be those who avoid all the obstacles and have bounded speed

$$\mathcal{A}^{s,t}(x,y) = \{\, \gamma\colon [s,t] \to X \mid \gamma(s) = x, \gamma(t) = y, \gamma \in \mathrm{AC}^{s,t}, \gamma(\theta) \in X_c(\theta), \\ |\gamma'|(\theta) \le v_m, \theta \in [s,t] \,\}. \tag{155}$$

Here $\mathrm{AC}^{s,t}$ denotes all the paths that are absolutely continuous. For any $\gamma \in \mathcal{A}^{s,t}(x,y)$, define its cost to be

$$J(\gamma) = \int_s^t L(\theta, \gamma, \gamma')\, \mathrm{d}\theta, \tag{156}$$

where L is the running cost or Lagrangian whose detail will be discussed below. Our problem is to find the path that minimizes J, i.e., to find

$$\gamma^* = \mathrm{argmin}_{\gamma \in \mathcal{A}^{s,t}(x,y)}\, J(\gamma). \tag{157}$$

For convenience, denote

$$J\big((s,x),(t,y)\big) = J(\gamma^*). \tag{158}$$

In this paper, we make the following assumption of the Lagrangian L:

C0. $L(\theta, \gamma, \gamma') = L(|\gamma'|)$ where $L\colon \mathbf{R}^+ \to \mathbf{R}$ is an increasing and convex function.

In other words, the cost depends solely on the speed of the curve. The constraints on acceleration, for example, is out of the scope of this paper.

Remark 6 *When the underlying metric space is* $\mathbf{R}^n$*, problem* (157) *can be completely put into the framework of optimal control. More specifically, we are solving*

$$\min_{\gamma} J(\gamma) = \int_s^t L(\gamma(\theta), u(\theta), \theta)\,\mathrm{d}\theta,$$

where $\gamma(\theta)$ *is determined by the linear differential equation*

$$\dot{\gamma}(\theta) = u(\theta),$$

$$|u| \leq v_m,$$

$$\gamma(s) = x, \gamma(t) = y,$$

and satisfies the path constraint

$$\phi_i(\theta, \gamma(\theta)) \geq 0, \quad t \in [s,t], \quad i \leq N.$$

where we assume that X_c *can be represented by the level set function of the obstacles* $\phi(x,\theta) \geq 0$*.*

The next propositions establish the existence of the optimal path.

Proposition 25 $\mathcal{A}$ *is sequentially compact in itself.*

Proof 10 *By Arzela-Ascoli theorem, in order to show that* $\mathcal{A}^{s,t}(x,y)$ *is sequentially compact, it suffices to show that* $\mathcal{A}^{s,t}(x,y)$ *is equicontinuous. However, this is straightforward since for any* γ*,*

$$d(\gamma(\theta_1), \gamma(\theta_2)) \leq v_m(\theta_1 - \theta_2). \tag{159}$$

Now let γ_n *be a sequence in* $\mathcal{A}^{s,t}(x,y)$ *and* $\gamma_n \to \gamma$*. Since*

$$d(\gamma_n(\theta_1), \gamma_n(\theta_2)) \leq v_m, \tag{160}$$

taking limit, we have

$$d(\gamma(\theta_1), \gamma(\theta_2)) \leq v_m.$$

This implies $\gamma \in \mathcal{A}^{s,t}(x, y)$. *This concludes the proof.*

From the above proposition, we immediately have

Theorem 26 *There exist an optimal path* γ^* *such that*

$$\gamma^* = \text{argmin}_{\gamma \in \mathcal{A}^{s,t}(x,y)} J(\gamma). \tag{161}$$

Proposition 27 *Functional* J *is convex on* $\mathcal{A}^{s,t}(x, y)$.

Proof 11 *This is straightforward since*

$$\begin{aligned} J(\lambda\gamma_1 + (1-\lambda)\gamma_2) &= \int_s^t L(|\lambda\gamma_1'(\theta) + (1-\lambda)\gamma_2'(\theta)|)\,\mathrm{d}\theta \\ &\leq \int_s^t L(\lambda|\gamma_1'(\theta)| + (1-\lambda)|\gamma_2'(\theta)|)\,\mathrm{d}\theta \\ &\leq \int_s^t \lambda L(|\gamma_1'(\theta)|) + (1-\lambda)L(|\gamma_2'(\theta)|)\,\mathrm{d}\theta \\ &\leq \lambda J(\gamma_1) + (1-\lambda)J(\gamma_2). \end{aligned}$$

Although functional J is convex, the optimal path is not necessarily unique. This is because $\mathcal{A}^{s,t}(x, y)$ is not convex due to the presence of obstacles.

6.2 Separability of the Optimal Path

As we have seen in the previous section, Problem (157) is a standard optimal control problem, hence the method of evolving junctions can be applied provided the optimal path is separable. In this section, we will show that under certain conditions, this simple structure indeed holds.

For convenience of the exposition, denote $\widetilde{P}_k = \{\, (\theta, x) \mid x \in P_k, \theta \in [s, t] \,\}$ and $\partial\widetilde{P}_k = \{\, (\theta, x) \mid x \in \partial P_k, \theta \in [s, t] \,\}$. Also to make dependence of the admissible set

explicitly on the target set and maximal speed, define for a time-dependent space Z,

$$\mathcal{A}^{s,t}(x,y;v_m,Z) = \{\,\gamma\colon [s,t]\to X_c \mid \gamma(s)=x, \gamma(t)=y, \gamma\in \mathrm{AC}^{s,t}, \\ \gamma(\theta)\in Z(\theta), |\gamma'|(\theta)\le v_m, \theta\in[s,t]\,\}, \quad (162)$$

and for $\tilde{x}=(s,x), \tilde{y}=(t,y)$, denote

$$J_0(\tilde{x},\tilde{y}) = J_0\big((s,x),(t,y)\big) = \min_{\gamma\in\mathcal{A}^{s,t}(x,y;v_m,X)} J(\gamma),$$
$$J_k(\tilde{x},\tilde{y}) = J_k\big((s,x),(t,y)\big) = \min_{\gamma\in\mathcal{A}^{s,t}(x,y;v_m,\partial P_k)} J(\gamma).$$

In other words, J_0 is the minimal cost connecting $\tilde{x},\tilde{y}$ in the free space assuming no obstacles and J_k is the minimal cost connecting $\tilde{x},\tilde{y}$ on the boundary of P_k.

To show that the optimal path is separable, we impose the following conditions on the Lagrangian and the motion of obstacles:

C1. $L(x) = L_0(x^2)$ for some increasing and real analytic function L_0. In addition, $L_0'(x) > 0$ and $L_0'(x) + 2L_0''(x)x > 0$.

C2. The boundary of each obstacle can be represented by its time-dependent level set function $\phi_k(t,x), t\in\mathbf{R}^+, x\in\mathbf{R}^2$. In addition, $\frac{\partial\phi_k}{\partial x}$ and $\frac{\partial\phi_k}{\partial\theta}$ are one-side real analytic.

Proposition 28 *If the Lagrangian is convex and increasing, then in an environment with no obstacles*

$$J_0\big((s,x),(t,y)\big) = \begin{cases} (t-s)L\left(\frac{\|x-y\|}{t-s}\right), & \frac{\|x-y\|}{t-s}\le v_m; \\ \infty, & \textit{otherwise.} \end{cases} \quad (163)$$

Proof 12 *Convexity of L implies that*

$$\frac{1}{t-s}\int_s^t L(|\dot\gamma(\theta)|)\,\mathrm{d}\theta \ge L\left(\frac{1}{t-s}\int_s^t |\dot\gamma(\theta)|\,\mathrm{d}\theta\right) \\ \ge L\left(\frac{\|x-y\|}{t-s}\right).$$

Therefore, the lower bound of the minimal cost is $(t-s)L(\frac{\|x-y\|}{t-s})$. *The first equality holds if* $|\dot{\gamma}|$ *is a constant or* L *is linear. Thus, if* L *is not linear, the minimal cost can be obtained only when* $|\dot{\gamma}(\theta)| = \frac{\|x-y\|}{t-s}$ *is a constant. This means that one must always move with constant speed. On the other hand, if* L *is linear, the equality holds only when* $\gamma(\theta)$ *travels along the line segment connecting* x *and* y*, the speed doesn't affect the minimal cost, which is still* $(t-s)L(\frac{\|x-y\|}{t-s})$.

Remark 7 *1. Condition (C0) is necessary to prevent infinite number of the minimal cost path as we will show below shortly.*

2. The linear Lagrangian $L(x) = \lambda x + 1 - \lambda$ *is ruled out because (1) it allows infinitely many solutions, (2)* $L(|p|)$ *is not* C^1 *at the origin.*

3. If $L(x) = \sqrt{x^2+1}$*, then the minimal cost is the shortest distance between* $\tilde{x} = (s,x)$ *and* $\tilde{y} = t, y)$ *in* $\widetilde{X}$.

4. The analyticity of L *and* ϕ *are technical conditions for the proof.*

One immediate corollary of proposition 28 is the following

Corollary 29 *The minimal cost path connecting* $\tilde{x} = (s,x)$ *and* $\tilde{y} = (t,y)$ *in* $\widetilde{X}$ *is the straight line segment connecting them.*

Remark 8 *Notice*

$$d_{\widetilde{X}}\big((s,x),(t,y)\big) = J_0\big((s,x),(t,y)\big) \tag{164}$$

defines a metric on space $\widetilde{X}$. *However,* $\widetilde{X}$ *endowed with this distance is not a metric space. This is because any two points in* $\widetilde{X}$ *have a natural order by the time component. In other words, one can not go from the future to the past.* $d_{\widetilde{X}}$ *is in fact quasimetric. Despite of the lack of symmetry, the triangle inequality still holds, which is the key to the separability of the minimal cost path.*

As mentioned above, if the Lagrangian L is not increasing, then there would be infinitely many of minimal cost paths as shown in the following example.

Example. Let $v_m = 1$ and $L\colon [0,1] \to \mathbf{R}$ be convex and have minimum at $v_0 = 1/2$. Then for a fixed starting point x_0,

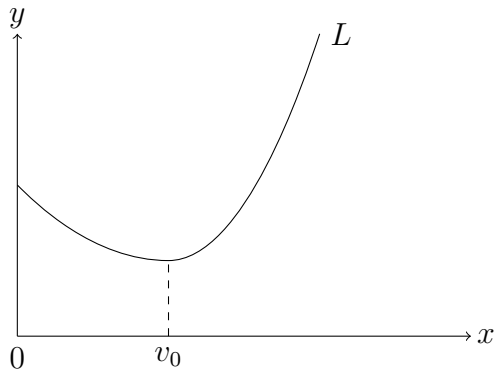


Figure 25: Lagrangian L.

$$J(t,x) = J\big((0,x_0),(t,x)\big) = \begin{cases} tL(\frac{1}{2}), & \frac{t}{2} \geq |x-x_0|; \\ tL(\frac{|x-x_0|}{t}), & \frac{t}{2} \leq |x-x_0|. \end{cases}$$

Hence the optimal time to reach x from x_0 is

$$J(x) = \min_t J(t,x) = \min_s |x-x_0| w L(\frac{1}{w}) = |x-x_0| w^* L(\frac{1}{w^*})$$

for some value w^*.

Consider an obstacle represented by $[-\epsilon,\epsilon]\times[-K,K]$ and starting point at $(-\epsilon,0)$ and ending point at $(\epsilon,0)$. Assume the obstacle moves along positive y direction with speed u. Then the minimal cost of the path, as a function of where one hits the lower corner of the obstacle, would be

$$\begin{aligned} F(x) &= J(\frac{K-x}{u}, x) + w^* L(\frac{1}{w^*})x \\ &= \begin{cases} \frac{K-x}{u} L(\frac{1}{2}) + w^* L(\frac{1}{w^*})x, & \frac{K-x}{u} \geq 2x; \\ \frac{K-x}{u} L(\frac{xu}{L-x}) + w^* L(\frac{1}{w^*}), & \frac{K-x}{u} \leq 2x. \end{cases} \end{aligned}$$

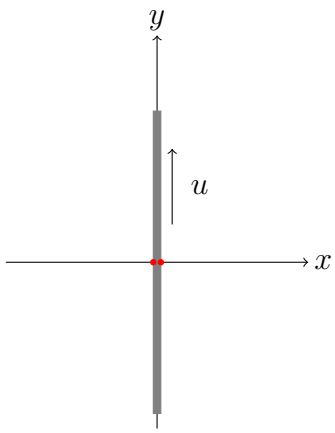


Figure 26: Obstacle moving with constant speed u.

The derivative of $F(x)$ is

$$F'(x) = \begin{cases} w^* L(\frac{1}{w^*}) - \frac{1}{u} L(\frac{1}{2}), & x \leq \frac{K}{2u+1}; \\ -\frac{1}{u} L(\beta) + (1 + \frac{\beta}{u}) L'(\beta) + w^* L(\frac{1}{w^*}), & \frac{K}{2u+1} \leq x \leq \frac{K}{u+1}, \beta = \frac{xu}{K-x} \geq \frac{1}{2}. \end{cases}$$

Now it is easy to see that if u is large enough, we will have $F'(x) > 0$. Hence $F(x)$ obtains its minimum when $x = 0$. In other words, the robot will stay at the origin until the obstacle clears its way. In this case, there are infinitely many ways for the robot to hang around with speed $1/2$ for time K/u.

Now we start to prove our main theorem, namely

Theorem 30 *The optimal path of problem (157) with $X = \mathbf{R}^2$ under conditions (C0), (C1) and (C2) is separable.*

The proof parallels with the one in [17]. The claim is validated by defining an approximate Lagrangian L_a of L such that it is well defined for all paths in $\widetilde{X}$ and it's arbitrarily large inside the obstacle. We show that the minimal cost path can be arbitrarily close to the optimal path of the original problem, while the structure of the approximate optimal path is known.

First, we establish the separability under the assumption that $\widetilde{P}_k$ is convex. This special case requires no analyticity of L or ϕ.

Proposition 31 *Let $\tilde{x} = (s, x), \tilde{y} = (t, y) \in \partial\widetilde{P}_k$, then the optimal path connecting $\tilde{x}, \tilde{y}$ lies on $\partial\widetilde{P}_k$ entirely if $\widetilde{P}_k$ is convex as a subset in $\widetilde{X}$.*

Proof 13 *Let $\gamma^*(\tilde{x}, \tilde{y})$ be the minimal cost curve. Function*

$$h(\theta) = \phi_k(\theta, \gamma(\theta)), \quad \theta \in [s, t] \tag{165}$$

is a continuous function. Assume on the contrary that

$$m = \max_{\theta\in[s,t]} h(\theta) > 0.$$

Let $\theta_0 = \operatorname{argmax} h(\theta)$ and

$$\theta_1 = \max\{\, \theta \mid \overline{\tilde{\gamma}(\theta)\tilde{\gamma}(\theta_0)} \in \widetilde{X}_c \,\},$$
$$\theta_2 = \min\{\, \theta \mid \overline{\tilde{\gamma}(\theta)\tilde{\gamma}(\theta_0)} \in \widetilde{X}_c \,\}.$$

Here $\overline{\tilde{x}\tilde{y}}$ denotes the straight line connecting $\tilde{x}, \tilde{y}$ in $\widetilde{X}$. Let $\tilde{x}_3 = (\theta_3, x_3)$ be the point on line $\overline{\tilde{\gamma}(\theta_1)\tilde{\gamma}(\theta_0)}$ that is closest to $\tilde{x}_0 = \tilde{\gamma}(\theta_0)$ and $\partial\widetilde{P}_k$ and $\tilde{x}_4 = (\theta_4, x_4)$ be the point on line $\overline{\tilde{\gamma}(\theta_0)\tilde{\gamma}(\theta_4)}$ that is closest to $\tilde{x}_0$ and $\partial\widetilde{P}_k$. Thus

$$\gamma^*(\tilde{x}_3, \tilde{x}_0) = \overline{\tilde{x}_3\tilde{x}_0},$$
$$\gamma^*(\tilde{x}_0, \tilde{x}_4) = \overline{\tilde{x}_0\tilde{x}_4}.$$

Notice that both $\gamma^*(\tilde{x}_3, \tilde{x}_0)$ and $\gamma^*(\tilde{x}_0, \tilde{x}_4)$ are both admissible. Moreover, we actually have $\tilde{x}_3, \tilde{x}_0, \tilde{x}_4$ on the same line segment in $\widetilde{X}$, otherwise by the triangular inequality, one can decrease the cost which contradicts with the optimality of $\gamma^*(x, y)$. By the convexity of $\widetilde{X}_c$, one must have $\tilde{x}_0 \in \widetilde{X}_c$. This contradicts with the selection of $\tilde{x}_0$.

Proposition 32 *Given two points $\tilde{x}, \tilde{y} \in \widetilde{X}_c$, if $\gamma^*(\tilde{x}, \tilde{y}) \in \widetilde{X}_c$, then $\gamma^*(\tilde{x}, \tilde{y}) = \overline{\tilde{x}\tilde{y}}$.*

Proof 14 *Let's consider the optimal control problem*

$$\dot{\gamma} = u,$$
$$J(\gamma) = \int_s^t L(|u(\theta)|)\,\mathrm{d}\theta.$$

Notice the speed constraint is not imposed at this stage. The Hamiltonian is

$$H(u) = L(|u|) + \lambda u.$$

Therefore, by Pontryagin Principle, the optimal control satisfies

$$\frac{\partial H}{\partial u} = 0,$$
$$\dot{\lambda} = -\frac{\partial H}{\partial x} = 0.$$

The second condition implies $\lambda = c$ is constant over $[s,t]$. In addition

$$\frac{\partial H}{\partial u} = L'(|u|)\frac{u}{|u|} + \lambda = 0 \tag{166}$$

yields that $|u|$ is also constant because L' is an increasing function. Therefore, by (166)*, u is constant over $[s,t]$. In other words, $\gamma^*(\tilde{x},\tilde{y})$ is straight line in $\widetilde{X}_c$.*

Proposition 31 and 32 implies that the optimal path intersects with each obstacle at most twice. In other words, we have

Proposition 33 *The optimal path of problem 157 with $X = \mathbf{R}^2$ under conditions (C0) and (C1) is separable provided further that each $\widetilde{P}_k$ is convex.*

Next we will show that for non-convex obstacles $\widetilde{P}_k$, the separability still holds. Let $g\colon \mathbf{R} \to \mathbf{R}$ be a continuous function such that

$$g(x) = \begin{cases} B, & x < -\epsilon; \\ \text{smooth and decreasing}, & -\epsilon \le x < 0; \\ 1, & x >= 0. \end{cases} \tag{167}$$

Here B is a large number which will be determined later. The approximate Lagrangian is defined as follows

$$L^\epsilon(t,x,v) = g(\phi(t,x))L(v). \tag{168}$$

In other words, we allow the feasible curve to enter the obstacle, but with a very large penalty. The set of admissible paths now becomes $\mathcal{A}^{s,t}(x, y; v_m, X)$. For convenience, we augment a path γ in $\mathcal{A}^{s,t}(x, y; v_m, X)$ to $\widetilde{X}$ by defining

$$\tilde{\gamma}(t) = (t, \gamma(t)) \tag{169}$$

and let $\tilde{\mathcal{A}}^{s,t}(x, y; v_m, X)$ be set of all augmented paths. Also define

$$J^{\epsilon}(\gamma) = \int_s^t L_{\epsilon}(\gamma)\, \mathrm{d}\theta \tag{170}$$

and

$$\gamma_{\epsilon}^* = \mathrm{argmin}_{\gamma \in \mathcal{A}^{s,t}(x,y;v_m,X)}\, J^{\epsilon}(\gamma).$$

Given a fixed time t, for each point x on $\partial P_k(t)$, we can associate it with a point inside $P_k(t)$ in the normal line with a distance of ϵ to x. All those points form another curve. Denote the domain enclosed by such curves by $P_k^{\epsilon}(t)$. Similarly, one can define $P_k^{2\epsilon}(t)$ and $P_k^{2\epsilon}(t) \subset P_k^{\epsilon}(t) \subset P_k(t)$. We have the following lemma

Lemma 34 *There exists a constant B in* (167) *such that $\tilde{\gamma}_{\epsilon}^*$ is entirely outside $\widetilde{P}_k^{2\epsilon}$ for each k.*

Proof 15 *For any two points $\tilde{x}_1 = (\theta_1, x_1), \tilde{x}_2 = (\theta_2, x_2) \in \partial \widetilde{P}_k^{\epsilon}$, denote $\overline{\tilde{x}_1 \tilde{x}_2}$ the line segment connecting them in $\tilde{X}$. Let*

$$S = \{\, (\tilde{x}_1, \tilde{x}_2) \mid \overline{\tilde{x}_1 \tilde{x}_2} \in \tilde{\mathcal{A}}^{s,t}(x, y; v_m, X), \overline{\tilde{x}_1 \tilde{x}_2} \cap \widetilde{P}_k^{2\epsilon} \neq \emptyset \,\}.$$

It is easy to see that S is compact and the function $h(\tilde{x}_1, \tilde{x}_2) = J(\overline{\tilde{x}_1 \tilde{x}_2})$ is always positive on S. Therefore, h has a positive minimum l. Now select an arbitrary path in $\tilde{\mathcal{A}}^{s,t}(x, y; v_m, X)$ whose length is L. Then any B such that $lB > L$ is the desired value.

Lemma 34 indicates that the optimal path under the new Lagrangian is close to being separable. We will construct this separable path γ_{ϵ} from $\gamma_{\epsilon}^*(t)$ in the following

way: if γ_ϵ^* is outside an obstacle, we keep it unchanged; if $\gamma_\epsilon^*(\theta)$ is inside an obstacle P_k, we project it on to ∂P_k. In other words,

$$\gamma_\epsilon^*(\theta) = \gamma_\epsilon(\theta) + \epsilon(\theta)\mathbf{n}(\theta), \tag{171}$$

where $\epsilon(t) < 2\epsilon$ and $\mathbf{n}$ is the normal to ∂P_k at $\gamma_\epsilon(\theta)$. From the construction, we have the following estimate

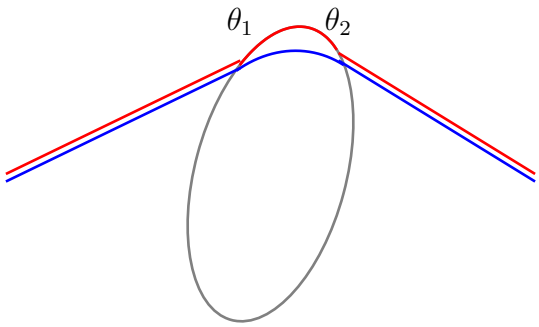


Figure 27: Red curve is γ_ϵ and blue curve is γ_ϵ^*.

Lemma 35 *Let $[\theta_1, \theta_2]$ be a continuous time interval that $\gamma_\epsilon^*(\theta)$ is inside one obstacle P_k, we have*

$$\left|\frac{d\gamma_\epsilon^*}{d\theta}\right| \geq (1 - 4\epsilon\kappa)\left|\frac{d\gamma_\epsilon}{d\theta}\right| \tag{172}$$

for some constant κ.

Proof 16 *From* (171), *we have*

$$\frac{d\gamma_\epsilon^*}{d\theta} = \frac{d\gamma_\epsilon}{d\theta} + \epsilon(\theta)\frac{d\mathbf{n}(\theta)}{d\theta} + \epsilon'(\theta)\mathbf{n}(\theta) \tag{173}$$

and hence

$$\begin{aligned}
\left|\frac{d\gamma_\epsilon^*}{d\theta}\right|^2 &= |\gamma_\epsilon'(\theta)|^2 + \epsilon(\theta)^2|\mathbf{n}'(\theta)|^2 + \epsilon'(\theta)^2 + 2\epsilon(\theta)(\gamma_\epsilon'(\theta), \mathbf{n}'(\theta)) \\
&\geq |\gamma_\epsilon'(\theta)|^2 + \epsilon(\theta)^2\kappa(\theta)^2|\gamma_\epsilon'(\theta)|^2 - 2\epsilon(\theta)\kappa(\theta)|\gamma_\epsilon'(\theta)|^2 \\
&= |\gamma_\epsilon'(\theta)|(1 - \epsilon(\theta)^2 - 2\epsilon(\theta)\kappa(\theta)) \\
&\geq |\gamma_\epsilon'(\theta)|(1 - 4\epsilon\kappa)^2.
\end{aligned}$$

where κ is chosen to be the maximal curvature. Therefore, we have

$$\left|\frac{d\gamma_\epsilon^*}{d\theta}\right| \geq (1-4\epsilon\kappa)\left|\frac{d\gamma_\epsilon}{d\theta}\right|.$$

The above lemma actually shows that

Corollary 36 *If $\gamma_\epsilon^* \in \mathcal{A}^{s,t}(x,y;v_m;X)$, then $\gamma_\epsilon \in \mathcal{A}^{s,t}(x,y;\frac{v_m}{1-4\epsilon\kappa};X)$.*

Next, we will show that γ_ϵ is separable. In other words, the intersections of γ_ϵ^* with each obstacle P_k is finite.

Proposition 37 *For ϵ sufficiently small, the number of intersections of γ_ϵ^* with obstacles is finite.*

Proof 17 *It is well known that $\alpha = \gamma_\epsilon^*$ satisfies the following Euler-Lagrange equation*

$$\frac{d}{dx}L^\epsilon(\theta,\alpha,\alpha') - \frac{d}{d\theta}(\frac{d}{dv}L^\epsilon(\theta,\alpha,\alpha')) = 0. \tag{174}$$

The Hessian of $L^\epsilon(p)$ is

$$L^\epsilon_{pp} = 2g_\epsilon(\phi(t,x))\begin{pmatrix} 2L_0''(p^2)p_1^2 + L_0'(p^2) & 2L_0''(p^2)p_1p_2 \\ 2L_0''(p^2)p_1p_2 & 2L_0''(p^2)p_2^2 + L_0'(p^2)\end{pmatrix}, \tag{175}$$

whose determinant is

$$\det L^\epsilon_{pp} = L_0'(p^2)(L_0'(p^2) + 2L_0''(p^2)p^2) > 0, \quad \textit{for all } p. \tag{176}$$

Therefore α is at least C^2 (See for example [34]) and we have

$$\alpha'' = (L^\epsilon_{pp}(\alpha'))^{-1}\left(-L^\epsilon_{px}(\alpha') - L^\epsilon_{pt}(\alpha') + L^\epsilon_x(\alpha')\right). \tag{177}$$

Under condition (C1) and (C2), $\alpha(t)$ is one-side analytic. As a result, $\phi(t,\alpha(t))$ is an one-side analytic function. Thus, it has finitely many roots and the number is independent of ϵ. In other words, γ_ϵ is separable.

As a result, for each ϵ, we can find one path γ_ϵ such that

$$\gamma_\epsilon \in \mathcal{A}^{s,t}(x, y; \frac{v_m}{1 - 4\epsilon\kappa}\epsilon, X), \tag{178}$$

$$d(\gamma_\epsilon, \gamma_\epsilon^*) < 2\epsilon. \tag{179}$$

Let $\mathcal{A} = \mathcal{A}^{s,t}(x, y; 2v_m, X)$, then $\gamma_\epsilon \in \mathcal{A}$ for sufficiently small ϵ. Consider the sequence $\gamma_n = \gamma_{\frac{1}{n}}, n \geq 1$. By taking a subsequence, we can assume $\gamma_n \to \gamma_0$. Also by taking the subsequence, let

$$\beta(\theta) = \liminf_{n\to\infty} \gamma_n'(\theta),$$

and

$$\gamma_1(\theta) = \int_s^\theta \beta(\theta)\,\mathrm{d}\theta.$$

Then we have

$$\begin{aligned}
|\gamma_1(\theta) - \gamma_0(\theta)| &= \left| \int_s^\theta \beta(u)\,\mathrm{d}u - \gamma_0(\theta) \right| \\
&= \left| \int_s^\theta \liminf \gamma_n'(u)\,\mathrm{d}u - \gamma_0(\theta) \right| \\
&\leq \liminf \left| \int_s^\theta \gamma_n'(u)\,\mathrm{d}u - \gamma_0(u) \right| \\
&= \liminf |\gamma_n(u) - \gamma_0(u)| \\
&= 0.
\end{aligned}$$

We now show that γ_0 is the desired optimal separable path.

Proposition 38 $\gamma_0 \in \mathcal{A}^{s,t}(x, y; v_m, X)$.

Proof 18 *It suffices to show the speed of γ_0 is bounded above by v_m. However, we have*

$$d(\gamma_n(\theta_1), \gamma_n(\theta_2)) \leq \frac{v_m}{1 - 4\kappa/n}|\theta_1 - \theta_2|. \tag{180}$$

Taking limit, we have

$$d(\gamma_0(\theta_1), \gamma_0(\theta_2)) \leq v_m|\theta_1 - \theta_2|. \tag{181}$$

This concludes the proof.

Proposition 39 *γ_0 is the minimal cost path.*

Proof 19 *First, for sufficiently small ϵ and by the boundedness of $\gamma_\epsilon)'$, we have*

$$L((1-4\epsilon\kappa)x) \geq L(x) - D\epsilon$$

for some constant D. Therefore,

$$\begin{aligned} J(\gamma_\epsilon) &= \int_s^t L(|\gamma_\epsilon'|)\,\mathrm{d}\theta \\ &\geq \int_s^t L((1-4\epsilon\kappa)|\gamma_\epsilon'|)\,\mathrm{d}\theta \\ &\geq \int_s^t L(|\gamma_\epsilon'|) - D\epsilon\,\mathrm{d}\theta \\ &= J(\gamma_\epsilon) - D\epsilon(t-s). \end{aligned}$$

Therefore,

$$J(\gamma^*) = J^\epsilon(\gamma^*) \geq J^\epsilon(\gamma_\epsilon) \geq J(\gamma_\epsilon) \geq J(\gamma_\epsilon) - D(t-s)\epsilon. \tag{182}$$

Taking limit, we have

$$J(\gamma_0) = J(\gamma^*). \tag{183}$$

Corollary 40 *The optimal path continuously depends on the maximal speed. More specifically, let $\gamma^*(v) = \mathrm{argmin}_{\gamma\in\mathcal{A}(x,y;v,X_c)} J(\gamma)$, then if v_n is decreasing and $v_n \to v_0$, we have*

$$\lim_n J(\gamma^*(v_n)) = J(\gamma^*(v_0)).$$

Proof 20 *This is a direct application* (182) *since*

$$J(\gamma^*(v_n)) \geq J(\gamma^*(v_0)) \geq J(\gamma_\epsilon) - D(t-s)\epsilon \geq J(\gamma^*(v_n)) - D(t-s)\epsilon.$$

Here $\epsilon = (1-v_0/v_n)/4\kappa$.

Theorem 41 *γ_0 is separable.*

Proof 21 *Let m be the upper bound of the number of junctions for all γ_n as in . Denote*

$$\mathbf{N}^m = \{\, (k_1, k_2, \cdots, k_p) \mid p \le m, k_i \le N \,\}$$

For each $\mathbf{T} = (k_1, \cdots, k_p) \in \mathbf{N}^m$, let

$$H(\mathbf{T}) = H(k_1, \cdots, k_p) = \partial P_{k_1} \times \cdots \times \partial P_{k_p}.$$

$$\mathcal{B}(T) = \mathcal{A}(x, y; v_m, X_c) \cap H(\mathbf{T}).$$

Now let γ^{**} solves

$$\min_{\mathbf{T} \in \mathbf{N}^m} \min_{\gamma \in \mathcal{B}(\mathbf{T})} J(\gamma).$$

Then by the Corollary 40, one must have $\gamma^{**} = \gamma_0$. Hence γ_0 is separable.

6.3 Optimal Path with Moving Obstacles

In this section, we will apply the method of evolving junctions to find the minimal cost path in an environment where obstacles are polygons and moving with constant speed. For convenience of exposition, we will omit the subscript and use P to denote a general obstacle. Let $\alpha\colon [0, 2\pi] \to \mathbf{R}^2$ be a parametrization of the boundary of obstacle P. Assume P moves with speed $\mathbf{v}(t)$, then the parametrization of the moving P is

$$R(u, t) = \alpha(u) + \int_0^t \mathbf{v}(\theta)\, \mathrm{d}\theta.$$

Under the parametrization of each obstacle, the junctions can be represented by the time and its parameter, in other words,

$$\tilde{x} = (t, x) = (t, u) = \tilde{u}, \quad x = R(t, u). \tag{184}$$

For the rest of the section, we will represent the junctions by

$$(\tilde{u}_0, \tilde{u}_1, \cdots, \tilde{u}_n, \tilde{u}_{n+1}). \tag{185}$$

Recall in (28), for any two junctions $\tilde{x} = (s, x), \tilde{y} = (t, y)$,

$$J_0((s,x),(t,y)) = \begin{cases} (t-s)L\left(\frac{\|x-y\|}{t-s}\right), & \frac{\|x-y\|}{t-s} \le v_m; \\ \infty, & \text{otherwise.} \end{cases} \tag{186}$$

To compute J_c for obstacle P, recall that by definition

$$J_c((s,u_1),(t,u_2)) = \min_u \int_s^t L(|\dot{x}(\theta)|)\,\mathrm{d}\theta,$$

where

$$x(\theta) = \alpha(u(\theta)) + \int_0^\theta \mathbf{v}(r)\,\mathrm{d}r, \quad \theta \in [s,t], \quad u(s) = u_1, u(t) = u_2. \tag{187}$$

From now on, assume further $\mathbf{v}(t)$ is a constant. Denote

$$\begin{aligned} F(\theta) &= L(|\dot{x}|) \\ &= L(|\alpha'(u(\theta)u'(\theta)) + \mathbf{v}|) \\ &= L_1\left(u'^2(\theta) + |\mathbf{v}|^2 + 2\mathbf{v}\cdot\alpha'(u(\theta))u'(\theta)\right) \\ &= L_1\left(u'^2(\theta) + |\mathbf{v}|^2 + u'(\theta)\beta(u(\theta))\right) \\ &= L_2\left(u'^2(\theta) + \beta(u(\theta))u'(\theta)\right), \end{aligned}$$

where

$$L_1(x) = L(\sqrt{x}),$$
$$L_2(x) = L_1(x + |v|^2) = L(\sqrt{x + |\mathbf{v}|^2}),$$
$$\beta(u) = 2\alpha'(u)\cdot\mathbf{v}.$$

Since each P is a polygon, the above $\beta(u)$ is piecewise constant. Let $\{v_0, v_1, \cdots, v_{n-1}\}$ be the set of coordinates of vertices of P in clockwise direction. Let $l_k = \|v_1 - v_0\| + \cdots + \|v_k - v_{k-1}\|$ be the length from v_k to v_1 on the boundary.

$$\beta(u) = \beta_k = (v_{k+1} - v_k)\cdot\mathbf{v}, \quad u \in (l_k, l_{k+1}].$$

For any $u_1, u_2 \in (l_k, l_{k+1})$,

$$F(\theta) = g(u'(\theta)) = L_2(u'^2(\theta) + \beta_k u'(\theta)),$$
$$u'(\theta)^2 + \beta_k u'(\theta) \le v_{max}^2 - |v|^2 \iff c_l(\beta_k) \le u'(\theta) \le c_u(\beta_k),$$

where

$$c_l(\beta_k) = \frac{1}{2}\left(-\beta_k - \sqrt{\beta_k^2 + 4(v_{max}^2 - |v|^2)}\right),$$
$$c_u(\beta_k) = \frac{1}{2}\left(-\beta_k + \sqrt{\beta_k^2 + 4(v_{max}^2 - |v|^2)}\right).$$

So now we have

$$J_c((s, u_1), (t, u_2)) = \min \int_s^t g(w)\,\mathrm{d}\theta,$$
$$\dot{u} = w,$$
$$u(s) = u_1, u(t) = u_2,$$
$$c_l(\beta_k) \le w \le c_u(\beta_k).$$

To solve this optimal control problem, notice

$$H = g(w) + \lambda w + \mu(w - c_u),$$
$$\frac{\partial H}{\partial w} = g'(w) + \lambda + \mu = 0,$$
$$\dot{\lambda} = -H_u = 0 \to \lambda \text{ is constant},$$
$$\mu \begin{cases} \ge 0, & w = c_u; \\ = 0, & w \le c_u. \end{cases}$$

The above analysis shows that $w = c_u$ or $w = c_i \in [c_l, c_u]$. Let's assume $w = c_u$ for time r, then

$$J_c\big((s, u_1), (t, u_2)\big) = \min rg(c_u) + (t - s - r)g(c_i), \tag{188}$$
$$rc_u + (t - s - r)c_i = u_2 - u_1. \tag{189}$$

Now assume $u_1 \in (l_i, l_{i+1}], u_2 \in (l_j, l_{j+1}]$. Then the cost along the path from u_1 to u_2 counterclockwisely is

$$J_c^+\big((s,u_1),(t,u_2)\big) = \min_{t_1,t_2,\cdots,t_{j-i}} J_c\big((s,u_1),(t_1,l_{i+1})\big) + \sum_{k=1}^{j-i-1} J_c\big((t_k,l_{i+k}),(t_{k+1},l_{i+k+1})\big) + J_c\big((t_{j-i}l_j),(t,u_2)\big),$$

and the cost along the path from u_1 to u_2 clockwisely is

$$J_c^-\big((s,u_1),(t,u_2)\big) = \min_{t_1,t_2,\cdots,t_{i-j}} J_c\big((s,u_1),(t_1,l_i)\big) + \sum_{k=1}^{i-j-1} J_c\big((t_k,l_{i-k+1}),(t_{k+1},l_{i-k})\big) + J_c\big((t_{i-j},l_{j+1}),(t,u_2)\big).$$

Here the indices may wrap around (for example $i, i+1, \cdots, n, 1, \cdots, j$), so all the index manipulations are in modulo n sense. We then have

$$J_c\big((s,u_1),(t,u_2)\big) = \min\left\{J_c^+\big((s,u_1),(t,u_2)\big), J_c^-((s,u_1),(t,u_2)\big)\right\}.$$

Here we give an example in which the Lagrangian is quadratic. We will show how to compute J_k numerically.

Example 1

$$c(x) = x^2 + c,$$

$$g(x) = x^2 + \beta_k x + |v|^2 + c,$$

$$g'(x) = 2x + \beta_k x.$$

In this case, (188) becomes

$$\begin{aligned} J'(r) &= g(c_u) - g(c_i) + (t-s-r)g'(c_i)\frac{\partial c_i}{\partial r} \\ &= g(c_u) - g(c_i) + g'(c_i)(c_i - c_u) \\ &= c_u^2 - c_i^2 + \beta_k(c_u - c_i) + (2c_i + \beta_k)(c_i - c_u) \\ &= (c_u - c_i)^2 \geq 0. \end{aligned}$$

Therefore, $r = 0$ and for $u_1, u_2 \in (l_k, l_{k+1}]$,

$$J_c\big((s,u_1),(t,u_2)\big) = (t-s)g(\frac{u_2-u_1}{t-s}) \tag{190}$$

$$= \frac{(u_2-u_1)^2}{t-s} + \beta_k(u_2-u_1) + (|\mathbf{v}|^2+c)(t-s). \tag{191}$$

thus for $u_1 \in (l_i, l_{i+1}]$ and $u_2 \in (l_j, l_{j+1}]$ we have

$$J_c^+\big((s,u_1),(t,u_2)\big) = \frac{(l_{i+1}-u_1)}{(t_1-s)} + \sum_{k=1}^{j-i-1} \frac{(l_{i+k+1}-l_{i+k})^2}{t_{k+1}-t_k} + \frac{u_2-l_j}{t-t_{j-i}} + \beta_i(l_{i+1}-u_1)$$
$$+ \sum_{k=1}^{j-i-1} \beta_{i+k}(l_{i+k+1}-l_{i+k}) + \beta_j(u_2-l_j) + (|\mathbf{v}|^2+c)(t-s).$$

Let $\Delta_k = t_{k+1} - t_k$, then we need to optimize

$$f(\Delta_0, \cdots, \Delta_{j-i}) = \sum_{k=1}^{j-i-1} \frac{(l_{i+k+1}-l_{i+k})^2}{\Delta_k} + \frac{(l_{i+1}-u_1)^2}{\Delta_0} + \frac{(u_2-l_j)^2}{\Delta_{j-i}} \tag{192}$$

subject to

$$\Delta_0 \geq \frac{l_{i+1}-u_1}{c_u(\beta_i)}, \Delta_k \geq \frac{l_{i+k+1}-l_{i+k}}{c_u(\beta_{i+k})}, \Delta_{j-i} \geq \frac{u_2-l_j}{c_u(\beta_j)}.$$

Problem (192) can be computed either by NLP approaches (only local) or by recursion as described in the appendix.

Besides the visibility constraints as discussed in [17], there is a speed constraint in the dynamic environment. Define the speed constraint function as

$$S(\tilde{u}_i, \tilde{u}_{i+1}) = \begin{cases} \|x_{k_i}(\tilde{u}_i) - x_{k_{i+1}}(\tilde{u}_{i+1})\| - v_m(t_{i+1}-t_i), & i \text{ even}; \\ \\ \int_{u_i}^{u_{i+1}} \frac{1}{c(\beta(\theta))}\,\mathrm{d}\theta - (t_{i+1}-t_i), & i \text{ odd}. \end{cases} \tag{193}$$

So the constraints are

$$S(\tilde{u}_i, \tilde{u}_{i+1}) \leq 0, \quad V(\tilde{u}_i, \tilde{u}_{i+1}) = 0, \quad 0 \leq i \leq n. \tag{194}$$

Here is the complete algorithm for finding the optimal path in an environment with moving obstacles.

Optimal Path amid Moving Obstacles

Input: coordinates of vertices and velocity of obstacle P_k,

starting and ending points x and y,

Lagrangian L,

number of intermittent diffusion intervals m.

Output: The optimal set γ_{opt} of junctions.

---

1. Initialization. Find the initial path $\gamma^{(0)} = (\tilde{u}_0, \cdots, \tilde{u}_{n+1})$;
2. Select duration of diffusion $\Delta T_l, l \leq m$;
3. Select diffusion coefficients $\sigma_l, l \leq m$;
4. **for** $l = 1 : m$
5. $\quad \gamma^{(l)} = \gamma^{(0)}$;
6. $\quad$ **for** $j = 1 : \Delta T_l$
7. $\quad\quad$ Find $\nabla^c J(\gamma^{(l)})$ by solving quadratic programming (109)
   with speed constraints S and visibility constraints V;
8. $\quad\quad$ Update $\gamma^{(l)}$ according to (111) with $\sigma(t) = \sigma_l$;
9. $\quad\quad$ Remove junctions from or add junctions to $\gamma^{(l)}$ when necessary;
10. $\quad$ **end**
11. $\quad$ **while** $\|\nabla^c J(\gamma^{(l)})\| > \epsilon$
12. $\quad\quad$ Update $\gamma^{(l)}$ according to (111) with $\sigma(t) = 0$;
13. $\quad\quad$ Remove junctions from or add junctions to $\gamma^{(l)}$ when necessary;
14. $\quad$ **end**
15. **end**
16. Compare $J(\gamma^{(l)}), l \leq m$ and set $\gamma_{opt} = \text{argmin}_{l \leq m} J(\gamma^{(l)})$.

Notice in (109), f is chosen to be the cost

$$J(\tilde{u}_0, \cdots, \tilde{u}_{n+1}) = \sum_{1 \le i \le n, \text{odd}} J_0(\tilde{u}_{i-1}, \tilde{u}_i) + J_c(\tilde{u}_i, \tilde{u}_{i+1}) \tag{195}$$

which is a function from $\mathbf{R}^{2(n+2)}$ to $\mathbf{R}$. The constraints C is

$$C(\tilde{u}_0, \cdots, \tilde{u}_{n+1}) = \begin{pmatrix} S(\tilde{u}_0, \tilde{u}_1) \\ S(\tilde{u}_1, \tilde{u}_2) \\ \cdots \\ S(\tilde{u}_n, \tilde{u}_{n+1}) \\ V(\tilde{u}_0, \tilde{u}_1) \\ V(\tilde{u}_2, \tilde{u}_3) \\ \cdots \\ V(\tilde{u}_n, \tilde{u}_{n+1}) \end{pmatrix}, \tag{196}$$

which is a function from $\mathbf{R}^{2(n+2)}$ to $\mathbf{R}^{3/2n+2}$. ∇J and ∇C are both computed numerically.

6.4 Numerical Experiment

We give an example in which the environment contains 5 moving polygonal obstacles. The cost is chose to be

$$L(x) = x^2 + 10 \tag{197}$$

and the starting point is $[0, 1]$, the ending point is $[2, 1]$. The terminal time is unspecified. The algorithm finds two minimizers. The global one has cost 5.844533, while the local minimizer has cost 5.846041. See Figure (28) and (29) for the snapshots of the movement. Notice the global minimizer has 8 junctions while the local only has 4.

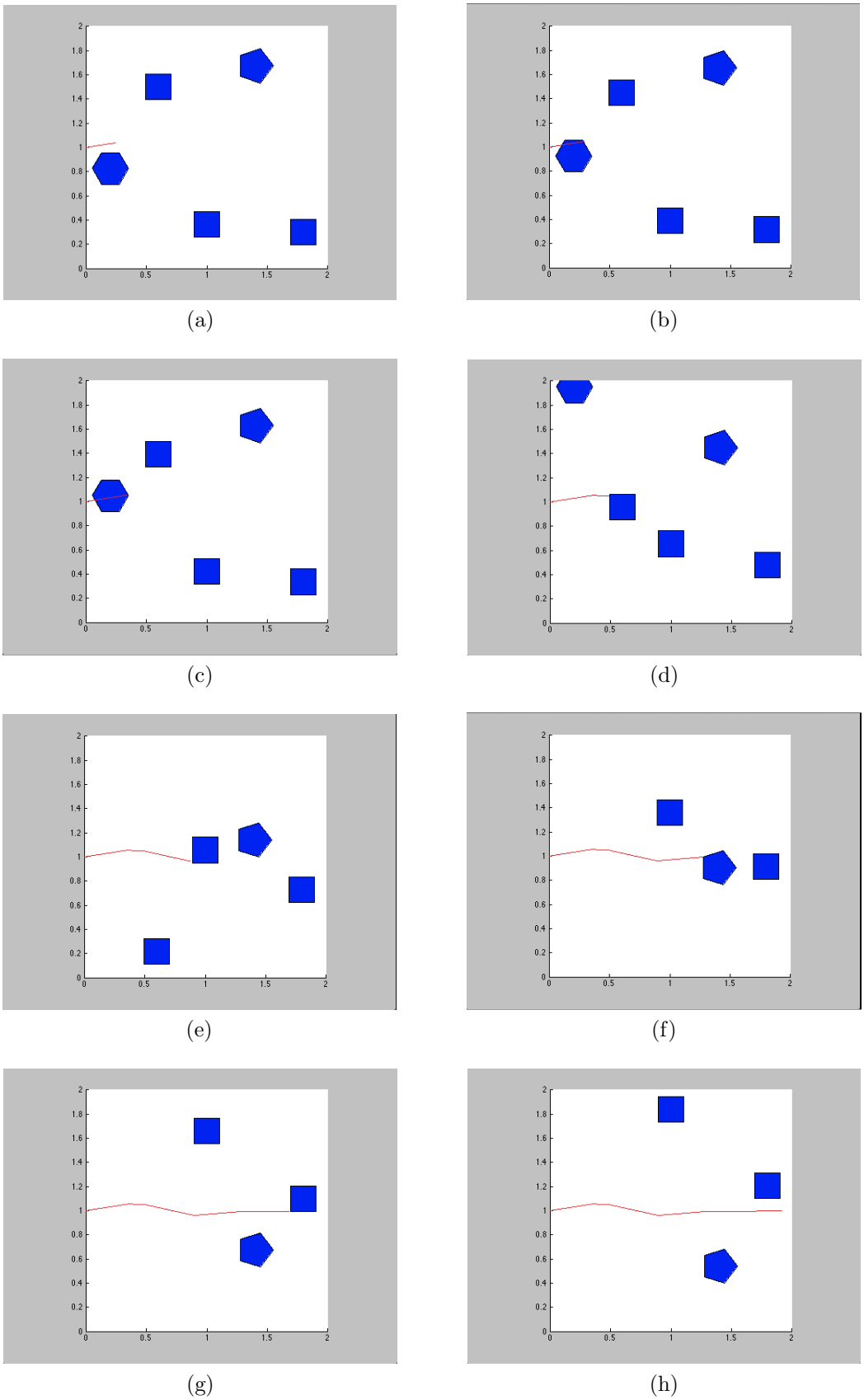


Figure 28: Snapshots of the global minimizer.

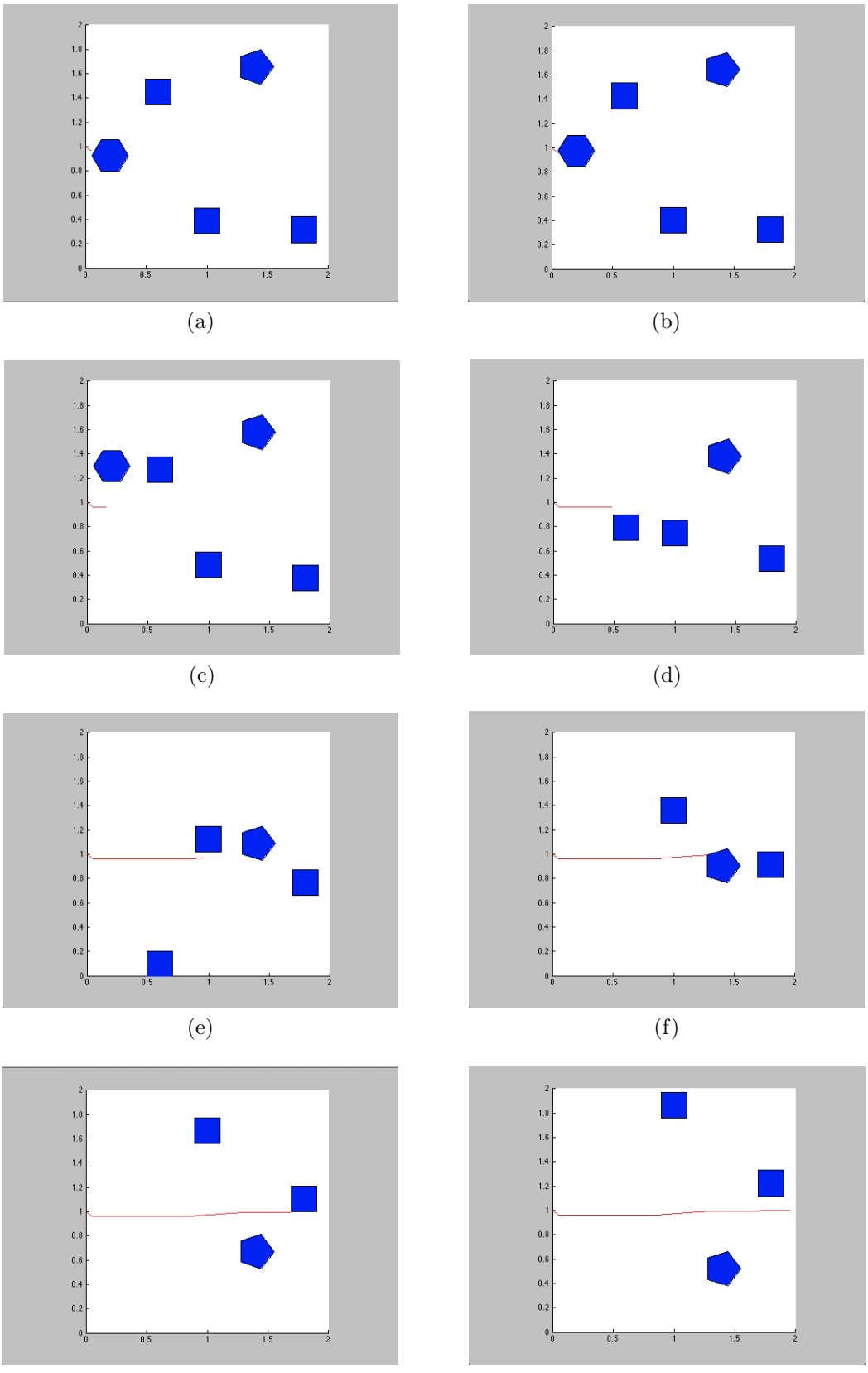


Figure 29: Snapshots of the local minimizer.

6.5 Appendix

6.5.1 Visibility Function of a Polygon

Given two points $\tilde{x} = (s, x)$ and $\tilde{y} = (t, y)$, the visibility function $V(\tilde{x}, \tilde{y})$ determines if moving from $\tilde{x}$ to $\tilde{y}$ with constant speed intersects with a given polygon P moving with speed w. Let $e = (u, v)$ be edges of P and O be a point inside the polygon. The line passing through $u = (u_1, u_2)$ and $v = (v_1, v_2)$ is

$$L(X, Y) = -(u_2 - v_2)X + (u_1 - v_1)Y - u_1v_2 + v_1u_2 = 0. \tag{198}$$

The signed distance from z to the line is

$$d_{u,v}(z) = -\operatorname{sign}(L(O))\frac{L(z)}{\|u - v\|}. \tag{199}$$

A point is inside the polygon if and only if the signed distance of z with respect to every edge is negative. As a result, the moving point

$$x(\theta) = x + \frac{\theta - s}{t - s}(y - x), \quad s \le \theta \le t \tag{200}$$

intersects with P if

$$d(\tilde{x}, \tilde{y}) = \min_{\theta} \max_{e_i} d_{u_i,v_i}(x(\theta)) < 0. \tag{201}$$

Notice that

$$\max_{e_i} d_{u_i(\theta),v_i(\theta)}(x(\theta)), \tag{202}$$

where

$$u_i(\theta) = u_i + \theta w,$$

$$v_i(\theta) = v_i + \theta w,$$

is a piecewise linear function and the minimizer can only be achieved at the intersections of $\{d_{u_i,v_i}\}_i$ as well as the ending points.

6.5.2 Recursive Method for Optimizing Problem (192)

Problem (192) can be simplified to the following problem

$$\min f(x_1, x_2, \cdots, x_n) = \sum_{i=1}^{n} \frac{c_i}{x_i} \tag{203}$$

subject to

$$\sum_{i=1}^{n} x_i = s, \quad x_i \geq d_i, \quad 1 \leq i \leq n. \tag{204}$$

Here $c_i > 0$ and $s \geq \sum_{i=1}^{n} d_i$. Let $J(c_1, \cdots, c_n; d_1, \cdots, d_n; s)$ denote the minimum of the above problem. By using Lagrange multiplier

$$H = \sum_{i=1}^{n} \frac{c_i}{x_i} - \mu(s - \sum_{i=1}^{n} x_i), \tag{205}$$

there is an interior critical point if

$$\frac{\partial H}{\partial x_i} = -\frac{c_i}{x_i^2} + \mu = 0,$$

$$x_i \geq d_i,$$

$$\sum_{i=1}^{n} x_i = s.$$

Solve for x_i, we have

$$x_i^* = s\frac{\sqrt{c_i}}{\sum_{i=1}^{n} \sqrt{c_i}}. \tag{206}$$

So if $x_i^* \geq d_i$ for any i, then

$$J(c_1, \cdots, c_n; d_1, \cdots, d_n; s) = f(x_1^*, \cdots, x_n^*). \tag{207}$$

Otherwise,

$$J(c_1, \cdots, c_n; d_1, \cdots, d_n; s) = \min_j \min f(x_1, \cdots, x_{j-1}, d_j, x_{j+1}, \cdots, x_n) \tag{208}$$

$$= \min_j J(c_1, \cdots, c_{j-1}, c_{j+1}, \cdots, c_n; d_1, \cdots, d_{j-1}, d_{j+1}, \cdots, c_n; s - d_j). \tag{209}$$

This is the recursion that can be used to compute J.

REFERENCES

[1] Aluffi-Pentini, F., Parisi, V., and Zirilli, F., "Global optimization and stochastic differential equations," *Journal of Optimization Theory and Applications*, vol. 47, no. 1, pp. 1–16, 1985.

[2] Ambrosio, L., Gigli, N., and Savaré, G., *Gradient flows: in metric spaces and in the space of probability measures*. Birkhauser, 2008.

[3] Arnold, A., Markowich, P., Toscani, G., and Unterreiter, A., "On convex sobolev inequalities and the rate of convergence to equilibrium for fokker-planck type equations," 2001.

[4] Balakrishnan, A. V. and Neustadt, L. W., *Computing methods in optimization problems: proceedings*. Academic Press, 1964.

[5] Bellman, R., "Dynamic programming and the smoothing problem," *Management Science*, vol. 3, no. 1, pp. 111–113, 1956.

[6] Bryson, A. E. and Ho, Y.-C., *Applied optimal control: optimization, estimation, and control*. Taylor & Francis Group, 1975.

[7] Canny, J. and Reif, J., "New lower bound techniques for robot motion planning problems," in *28th Annual Symposium on Foundations of Computer Science*, pp. 49–60, IEEE, 1987.

[8] Canny, J. F. and Lin, M. C., "An opportunistic global path planner," in *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*, pp. 1554–1559, IEEE, 1990.

[9] Chen, D. Z., "On the all-pairs euclidean short path problem," in *Proceedings of the sixth annual ACM-SIAM symposium on Discrete algorithms*, pp. 292–301, Society for Industrial and Applied Mathematics, 1995.

[10] Chen, D. Z., Das, G., and Smid, M., "Lower bounds for computing geometric spanners and approximate shortest paths," in *In Proc. 8th Canad. Conf. Comput. Geom*, Citeseer, 1996.

[11] Chiang, T.-S., Hwang, C.-R., and Sheu, S. J., "Diffusion for global optimization in r^n," *SIAM Journal on Control and Optimization*, vol. 25, no. 3, pp. 737–753, 1987.

[12] Choi, J., Sellen, J., and Yap, C., "Precision-sensitive euclidean shortest path in 3-space," in *Proceedings of the eleventh annual symposium on Computational geometry*, pp. 350–359, ACM, 1995.

[13] Choi, J., Sellen, J., and Yap, C.-K., "Precision-sensitive Euclidean shortest path in 3-space," in *Proceedings of the eleventh annual symposium on Computational geometry*, pp. 350–359, ACM, 1995.

[14] Choi, J., Sellen, J., and Yap, C.-K., "Approximate Euclidean shortest paths in 3-space," *International Journal of Computational Geometry & Applications*, vol. 7, no. 04, pp. 271–295, 1997.

[15] Chow, S.-N. and Hale, J. K., *Methods of bifurcation theory*, vol. 138. Springer-Verlag, 1982.

[16] Chow, S.-N., Yang, T.-S., and Zhou, H., "Global Optimizations by Intermittent Diffusion," *accepted by International Journal of Bifurcation and Chaos*, 2012.

[17] Chow, S.-N., Lu, J., and Zhou, H.-M., "Finding the shortest path by evolving junctions on obstacle boundaries (E-JOB): An initial value ODE's approach," *Applied and Computational Harmonic Analysis*, 2012.

[18] Chow, S.-N., Lu, J., and Zhou, H., "Fast numerical methods based on sdes for several problems related to the shortest path," *accepted to MAA special issue in honor of Stanley Osher on his 70th birthday.*

[19] Chow, S.-N., Lu, J., and Zhou, H., "Shortest path amid 3-d polyhedral obstacles," *submitted to SIAM scientific computing*, 2012.

[20] Clarkson, K., "Approximation algorithms for shortest path motion planning," in *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pp. 56–65, ACM, 1987.

[21] Dennis, J. E. and Schnabel, R. B., *Numerical methods for unconstrained optimization and nonlinear equations*, vol. 16. Society for Industrial and Applied Mathematics, 1987.

[22] Dias, R., Garcia, N., and Zambom, A., "Monte carlo algorithm for trajectory optimization based on markovian readings," *Computational Optimization and Applications*, vol. 51, no. 1, pp. 305–321, 2012.

[23] do Carmo, M., *Riemannian geometry*. Birkhauser, 1992.

[24] Fahimi, F., *Autonomous robots: modeling, path planning, and control*. Springer, 2008.

[25] Fletcher, R., "Practical methods of optimization," 1987.

[26] Gilbert, E., Johnson, D., and Keerthi, S., "A fast procedure for computing the distance between complex objects in three-dimensional space," *Robotics and Automation, IEEE Journal of*, vol. 4, no. 2, pp. 193–203, 1988.

[27] Gill, P. E., Murray, W., and Wright, M. H., "Practical optimization," 1981.

[28] Goroshin, R., Huynh, Q., and Zhou, H., "Approximate solutions to several visibility optimization problems," *Communications in Mathematical Sciences*, vol. 9, no. 2, pp. 535–550, 2011.

[29] Hart, P. E., Nilsson, N. J., and Raphael, B., "A formal basis for the heuristic determination of minimum cost paths," *Systems Science and Cybernetics, IEEE Transactions on*, vol. 4, no. 2, pp. 100–107, 1968.

[30] Hershberger, J. and Suri, S., "An optimal algorithm for Euclidean shortest paths in the plane," *SIAM J. Comput.*, vol. 28, no. 6, pp. 2215–2256, 1999.

[31] Holley, R. and Stroock, D., "Logarithmic sobolev inequalities and stochastic ising models," *Journal of statistical physics*, vol. 46, no. 5, pp. 1159–1194, 1987.

[32] Jiang, K., Seneviratne, L., and Earles, S., "Finding the 3d shortest path with visibility graph and minimum potential energy," in *Intelligent Robots and Systems' 93, IROS'93. Proceedings of the 1993 IEEE/RSJ International Conference on*, vol. 1, pp. 679–684, IEEE, 1993.

[33] Jiang, K., Seneviratne, L., and Earles, S., "Finding the 3D shortest path with visibility graph and minimum potential energy," in *Intelligent Robots and Systems' 93, IROS'93. Proceedings of the 1993 IEEE/RSJ International Conference on*, vol. 1, pp. 679–684, IEEE, 1993.

[34] Jost, J. and Li-Jost, X., *Calculus of variations*, vol. 64. Cambridge University Press, 1998.

[35] Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P., "Optimization by simulated annealing," *science*, vol. 220, no. 4598, p. 671, 1983.

[36] Koenig, S. and Likhachev, M., "Fast replanning for navigation in unknown terrain," *Robotics, IEEE Transactions on*, vol. 21, no. 3, pp. 354–363, 2005.

[37] Latombe, J.-C., *Robot motion planning*. Springer, 1990.

[38] LaValle, S. M., *Planning algorithms*. Cambridge Univ Pr, 2006.

[39] LaValle, S. M., "Rapidly-exploring random treesz: A new tool for path planning," 1998.

[40] Livne, E., Mineau, D., BRYSON, A., and DENHAM, W., "Optimal programming problems with inequality constraints. ii-solution by steepest-ascent," *AIAA Journal*, vol. 2, no. 1, pp. 25–34, 1964.

[41] Lu, J., Diaz-Mercado, Y., Egerstedt, M., Zhou, H., and Chow, S.-N., "Shortest paths through 3-dimensional cluttered environments," *accepted to International Conference on robotics and automation*, 2014.

[42] Lumelsky, V. J., "Algorithmic and complexity issues of robot motion in an uncertain environment," *Journal of Complexity*, vol. 3, no. 2, pp. 146–182, 1987.

[43] Markowich, P. A. and Villani, C., "On the trend to equilibrium for the fokker-planck equation: an interplay between physics and functional analysis," *Mat. Contemp*, vol. 19, pp. 1–29, 2000.

[44] Mc Reynolds, S. R. and Bryson Jr, A. E., "A successive sweep method for solving optimal programming problems.," tech. rep., DTIC Document, 1965.

[45] Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A., Teller, E., and others, "Equation of state calculations by fast computing machines," *The journal of chemical physics*, vol. 21, no. 6, p. 1087, 1953.

[46] Mitchell, J. S. B., "Shortest path and networks," *Handbook of Discrete and computational geometry*, pp. 755–778, 1997.

[47] Mitchell, J. S. B. and Sharir, M., "New results on shortest paths in three dimensions," in *Proceedings of the twentieth annual symposium on Computational geometry*, pp. 124–133, ACM, 2004.

[48] Mitchell, J. S. and Sharir, M., "New results on shortest paths in three dimensions," in *Proceedings of the twentieth annual symposium on Computational geometry*, pp. 124–133, ACM, 2004.

[49] Nocedal, J. and Wright, S. J., *Numerical optimization*. Springer Science+ Business Media, 2006.

[50] Osher, S. and Fedkiw, R. P., *Level set methods and dynamic implicit surfaces*. Springer Verlag, 2003.

[51] Osher, S. and Fedkiw, R., *Level set methods and dynamic implicit surfaces*, vol. 153. Springer, 2003.

[52] Papadimitriou, C., "An algorithm for shortest-path motion in three dimensions," *Information Processing Letters*, vol. 20, no. 5, pp. 259–263, 1985.

[53] Papadimitriou, C., "An algorithm for shortest-path motion in three dimensions," *Information Processing Letters*, vol. 20, no. 5, pp. 259–263, 1985.

[54] Pontriagin, L. S., *The mathematical theory of optimal processes*, vol. 4. CRC PressI Llc, 1962.

[55] Quinlan, S., "Efficient distance computation between non-convex objects," in *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*, pp. 3324–3329, IEEE, 1994.

[56] Quinlan, S. and Khatib, O., "Elastic bands: Connecting path planning and control," in *Robotics and Automation, 1993. Proceedings., 1993 IEEE International Conference on*, pp. 802–807, IEEE, 1993.

[57] Richards, A. and How, J., "Aircraft trajectory planning with collision avoidance using mixed integer linear programming," in *American Control Conference, 2002. Proceedings of the 2002*, vol. 3, pp. 1936–1941 vol.3, 2002.

[58] Risken, H., *The Fokker-Planck equation: methods of solution and applications*. Springer Verlag, 1985.

[59] Sabry Hassouna, M., Abdel-Hakim, A. E., and Farag, A. A., "PDE-based robust robotic navigation," *Image and Vision Computing*, vol. 27, no. 1-2, pp. 10–18, 2009.

[60] Schreiber, Y., *Euclidean shortest paths on polyhedra in three dimensions*. PhD thesis, Tel Aviv University, 2007.

[61] Schwartz, J. T., *Planning, geometry, and complexity of robot motion*. Ablex Pub, 1987.

[62] Sedeno-Noda, A. and Gonzalez-Martin, C., "An efficient label setting/correcting shortest path algorithm," *Computational Optimization and Applications*, vol. 51, no. 1, pp. 437–455, 2012.

[63] Sethian, J. A., "A fast marching level set method for monotonically advancing fronts," *Proceedings of the National Academy of Sciences*, vol. 93, no. 4, p. 1591, 1996.

[64] Sethian, J. A., *Level set methods: Evolving interfaces in geometry, fluid mechanics, computer vision, and materials science*, vol. 1999. Cambridge University Press Cambridge:, 1996.

[65] Speyer, J. L., Mehra, R. K., and Bryson Jr, A. E., "The separate computation of arcs for optimal flight paths with state variable inequality constraints," *Advanced Problems and Methods for space flight optimization*, pp. 53–68, 1969.

[66] Stentz, A., "Optimal and efficient path planning for partially-known environments," in *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*, pp. 3310–3317, IEEE, 1994.

[67] Stentz, A. and others, "The focussed D* algorithm for real-time replanning," in *International Joint Conference on Artificial Intelligence*, vol. 14, pp. 1652–1659, Lawrence Erlbaum Associates LTD, 1995.

[68] Szu, H. and Hartley, R., "Fast simulated annealing* 1," *Physics letters A*, vol. 122, no. 3-4, pp. 157–162, 1987.

[69] Tsai, Y. H. R., Cheng, L. T., Osher, S., Burchard, P., and Sapiro, G., "Visibility and its dynamics in a PDE based implicit framework* 1," *Jo issn=0178-4617*, 1992.

[70] Tsai, Y. H. R., Cheng, L. T., Osher, S., and Zhao, H. K., "Fast sweeping algorithms for a class of hamilton-jacobi equations," *SIAM journal on numerical analysis*, pp. 673–694, 2004.

[71] Van Den Berg, J. and Overmars, M., "Planning time-minimal safe paths amidst unpredictably moving obstacles," *The International Journal of Robotics Research*, vol. 27, no. 11-12, pp. 1274–1294, 2008.

[72] Vasile, M. and Locatelli, M., "A hybrid multiagent approach for global trajectory optimization," *Journal of Global Optimization*, vol. 44, no. 4, pp. 461–479, 2009.

[73] Yang, G. and Kapila, V., "Optimal path planning for unmanned air vehicles with kinematic and tactical constraints," in *Decision and Control, 2002, Proceedings of the 41st IEEE Conference on*, vol. 2, pp. 1301–1306 vol.2, 2002.

[74] Zhao, H., "A fast sweeping method for eikonal equations," *Mathematics of computation*, vol. 74, no. 250, pp. 603–628, 2005.

[75] Zhao, H., "A fast sweeping method for eikonal equations," *Mathematics of computation*, vol. 74, no. 250, pp. 603–628, 2005.